D T 02 A8601

31650T 02A 8601

CDC 31650T

MISSILE SYSTEMS MAINTENANCE SPECIALIST

(AFSC 41150B)

Volume 2A

Special Circuits and Digital Principles

AIR UNIVERSITY

Extension Course Institute
Air University

Prepared by
MSgt Robert L. Pieper

Reviewed by
Pamela G. Brown

Edited by
Beverly S. Barnes

3360 TECHNICAL TRAINING GROUP
3330 TECHNICAL TRAINING WING (ATC)
CHANUTE AIR FORCE BASE, ILLINOIS 61868-5000

EXTENSION COURSE INSTITUTE (AU)
GUNTER AIR FORCE STATION, ALABAMA 36118-5643

Preface

THIS VOLUME of CDC 31650T is the second of two volumes pertaining to electronic principles. This volume is divided into five chapters. Chapters 1, 2, and 3 are found in Module 10005, Digital Techniques, Units 1, 2, and 4 (which are included in your course package). Chapter 4 discusses complement arithmetic and its applications. Chapter 5 waveshaping and timing circuits to include differentiators, integrators, and timing distribution and control logic.

A glossary of terms used in this course is printed as a supplement. Use it as needed.

Code numbers appearing on figures are for preparing agency identification only.

The inclusion of names of any specific commercial product, commodity, or service in this publication is for information purposes only and does not imply indorsement by the Air Force.

To get an *immediate response* to your questions concerning subject matter in this course, call the author in AV 862-3519 between 0800 and 1600 (CT), Monday through Friday. Otherwise, write the author at 3360 TCHTG/TTGU-M, Chanute AFB, IL 61868–5000 to point out technical errors you find in the text, volume review exercises, or course examination. Sending subject matter questions to ECI slows response time.

NOTE: Do not use the Suggestion Program to submit changes to this course.

Consult your education officer, training officer, or NCO if you have questions on course enrollment or administration, Your Key to a Successful Course, and irregularities (possible scoring errors, printing errors, etc.) on the volume review exercises and course examination. Send questions these people can't answer to ECI, Gunter AFS AL 36118–5643, on ECI Form 17, Student Request for Assistance.

This volume is valued at 42 hours (14 points). These hours and points include those assigned to the modules you will study in Chapters 1, 2, 3.

Material in this volume is reviewed annually for technical accuracy, adequacy, and currency. For SKT purposes the examinee should check the Index of ECI Study Reference Material to determine the correct references to study.

Contents

Page

Preface .... *iii*

Chapter

1 Introduction to Digital Techniques and Numbering Systems (Module 10005) 1

2 Digital Logic and Boolean Algebra (Module 10005) 2

3 Sequential Logic (Module 10005) 3

4 Complement Arithmetic 4

5 Waveshaping Circuits 18

Answers for Exercises 33

Introduction to Digital Techniques and Numbering Systems (Module 10005)

NOTE: For objectives 200–219, study objectives 001–020 in Module 10005, *Digital Techniques,* Unit 1, which accompanies this volume. When you complete Unit 1 of Module 10005, return to the text.

Module 10005
Unit 1

CDC 31650T–2A Objectives	*Unit Objectives*
200	001
201	002
202	003
203	004
204	005
205	006
206	007
207	008
208	009
209	010
210	011
211	012
212	013
213	014
214	015
215	016
216	017
217	018
218	019
219	020

Digital Logic and Boolean Algebra (Module 10005)

NOTE: For objectives 220–243, study objectives 001–024 in Module 10005, *Digital Techniques,* Unit 2, which accompanies this volume. When you complete Unit 2 of Module 10005, return to the text.

Module 10005
Unit 2

CDC 31650T–2A Objectives	*Unit Objectives*
220	001
221	002
222	003
223	004
224	005
225	006
226	007
227	008
228	009
229	010
230	011
231	012
232	013
233	014
234	015
235	016
236	017
237	018
238	019
239	020
240	021
241	022
242	023
243	024

Sequential Logic (Module 10005)

NOTE: For objectives 244–259, study objectives 001–017 in Module 10005, *Digital Techniques,* Unit 1, which accompanies this volume. When you complete Unit 4 of Module 10005, return to the text.

Module 10005
Unit 4

CDC 31650T–4 Objectives	*Unit Objectives*
244	001
245	002
246	003
247	004
248	005
249	006
250	007
251	008
252	009
253	010
254	011
255	012
256	014
257	015
258	016
259	017

CHAPTER 4

Complement Arithmetic

THIS CHAPTER defines the radix and radix minus 1 complement of numbers and discusses how complement arithmetic is used to simplify computer mechanization. Machine-oriented methods using complement arithmetic to perform the basic arithmetic operations of addition, subtraction, multiplication, and division are demonstrated.

4-1. Basic Complement Arithmetic

A computer is usually designed to perform its arithmetic operations using either addition (only) or subtraction (only). Thus, in a given computer, all computations are performed by adding. In another machine, all operations are performed by subtracting. An additive computer addition and subtraction are both possible, since algebraically:

$a + b = s$ (equation 1)
and $a + (-b) = d$ (equation 2)

In equation 1, b is a positive quantity and a + b equals s (the sum). In equation 2, b is a negative quantity, and, when added, d represents the difference, or a − b. This procedure can be used in the computer only after a method is found for identifying and manipulating both positive and negative numbers, a process usually accomplished by using complement arithmetic.

260. Given numbers and their bases, supply complements.

Basic Complement Procedures. An arithmetical complement is defined as the difference between a number and the modulus of the system. The modulus of a system is the total number of discrete quantities representable in that system. Thus, in base ten:

2 is the complement of 8
$(10 - 8 = 2)$

26 is the complement of 74
$(100 - 74 = 26)$

744 is the complement of 256
$(1000 - 256 = 744)$

Referring back to the definition, it is seen that complement arithmetic is not limited to base ten. Thus, in base eight:

2 is the complement of 6
$(10 - 6 = 2)$

4 is the complement of 74
$(100 - 74 = 4)$

522 is the complement of 256
$(1000 - 256 = 522)$

The relationship between any number and its complement in any base may then be defined by the following equation:

$C = B^D - n$ (equation 3)

Where:

C = the complement
B = the base of the system being used, expressed in that system
D = the number of digits in n
n = any number

Observe that 2_{10} is the complement of 8_{10}, $(C = 10^1 - 8)$ and at the same time 2_{10} is a number that has 8_{10} $(C = 10^1 - 2)$ as a complement. Thus, developing a method of differentiating between 2 as a number and 2 as a complement of a number must also be accomplished before arithmetic operations can be performed using the complement method.

An interesting situation develops by modifying equation 3 as follows:

$C = B^{D+1} - n$ (equation 4)

Considering the previously developed complements and using equation 4 yields, in base ten:

92 is the complement of 8
$(10^{1+1} - 8 = 92)$

926 is the complement of 74
$(10^{2+1} - 74 = 926)$

9744 is the complement of 256
($10^{3+1} - 256 = 9744$)

and in base eight:

72 is the complement of 6
($100 - 6 = 72$)

704 is the complement of 74
($1000 - 74 = 704$)

7522 is the complement of 256
($10000 - 256 = 7522$)

Observe here that the complement is preceded in each case by the highest digit in the base used. If all numbers that are not complements are preceded by 0, the identification of complemented numbers is solved, removing the possible ambiguity that might develop in interpreting a positive 92_{10} and complement of 8_{10}.

Exercises (260):

1. 92_{10} is the complement of __________ .

2. 32_4 is the complement of __________ .

3. What is the complement of 250_{10}?

4-2. Radix Complement Arithmetic

It should be clear now that the complement of a number represents the negative value of that number. Thus, to perform a subtraction, it is necessary only to complement the subtrahend and continue as in normal addition. If, at any time, a resultant is a negative number, this fact will be indicated by the appearance of the complement form in the answer. Furthermore, the complement of a complement is the original number.

261. State characteristics of radix complements and solve problems using radix complements.

Radix Complement Procedures. Let a = 047 and a′ = 953 (read as "The complement of a = 953"), and let b = 023 and b′ = 977. There are eight possible combinations of a and b that can occur. These are a ± (+b); a ± (−b), −a ± (+b), and −a ± (−b).

Substituting the given values for a and b in each case, we have the following:

```
(1) a + (+b)          (2) a − (+b)

   047                   047      047
   023                  −023  =  +977
   070                   024  =  1024
```

TABLE 4-1

MACHINE REPRESENTATION OF +5 TO −5
(RADIX COMPLEMENT)

5	000	000	000	000	101
4	000	000	000	000	100
3	000	000	000	000	011
2	000	000	000	000	010
1	000	000	000	000	001
0	000	000	000	000	000
−1	111	111	111	111	111
−2	111	111	111	111	110
−3	111	111	111	111	101
−4	111	111	111	111	100
−5	111	111	111	111	011

```
(3) a + (−b)          (4) a − (−b)

   047                   047      047
   977                  −977  =  +023
  1024                            070

(5) −a + (−b)         (6) −a − (+b)

   953                   953      953
   023                  −023  =  +977
   976 = −24                     1930 = −70

(7) −a + (−b)         (8) −a − (−b)

   953                   953      953
   977                  −977  =   023
  1930                            976 = −24
```

Consideration of the preceding examples brings to light an interesting phenomenon: that of overflow, as typified by the 1 in the result of examples 2, 3, 6, and 7. Here the most significant digit (1) contributes nothing to the validity of the answer and is disregarded.

The preceding statements also apply to binary complement arithmetic. Thus, the complement is indicated when the most significant digit is 1, the highest permissible digit of the system, and a positive or uncomplemented number is indicated if the most significant digit is zero. The following discussion is limited to the use of a finite number of digits, since this is the situation actually prevailing within the computer. For simplicity, the binary numbers in this discussion are limited to 15 places. Let's look at the binary equivalent of some numbers in the vicinity of zero in table 4-1.

The representation in table 4-1 is slightly awkward but may be converted to the more conveniently handled octal form by grouping, giving:

5	00005
4	00004
3	00003
2	00002
1	00001
0	00000
−1	77777
−2	77776
−3	77775
−4	77774
−5	77773

Here each complement is developed by subtracting the absolute value of the number from 2^{15}.

Remembering that the only way to determine whether or not a number is positive or negative is by evaluating the most significant digit, we see that values from 00000 through 37777 (011 111 111 111 111) are positive and values from 40000 (100 000 000 000 000) through 77777 are negative (−40000 through −00001). However, since complementing a negative number gives a positive number, an undefined point exists, for complementing 40000 yields 40000, and the same number cannot be both positive and negative. To avoid this ambiguity, negative numbers should be limited to values between 40001 and 77777 (−37777 and −00001).

Now consider some sample problems:

(1) (+5) + (−4) = 1

$$\begin{array}{r} 00005 \\ \underline{77774} \\ \boxed{1}\,00001 \end{array}$$

Once again, this overflow is dropped.

(2) (−6) + (−6) = −14

$$\begin{array}{r} 77772 \\ \underline{77772} \\ 77764 \doteq -4 \end{array}$$

(3) (−12) − (−6) = −14

$$\begin{array}{r} 77766 \\ \underline{77772} \\ 77764 \doteq -4 \end{array}$$

(4) (32000) + (32000) = 64000

$$\begin{array}{r} 32000 \\ \underline{32000} \\ 64000 \end{array}$$

100000 − 64000 = −1400

Note here that the addition has produced an overflow into the sign bit (highest order) position, thereby indicating a negative result.

Once again it is emphasized that the largest possible positive number the machine can contain as the result of a computation is 37777. This problem would have to be scaled (examined to determine if the result will produce an overflow) prior to entering it into the machine.

(5) (−37775) + (−37774) = −77771

$$\begin{array}{r} 40003 \\ \underline{40004} \\ \boxed{1}\,00007 = 7 \end{array}$$

Here again is an erroneous sum. The 1 is dropped and it appears that the sum is a positive 7. Once again, the limitations of the machine have been exceeded, since the largest negative number that it can properly define is 40001 (−37777). The solution is found in proper scaling of the factors.

The important facts about the radix complement arithmetic method are:

a. The complement is indicated by the presence of the highest digit of the base being used in the most significant digit's position.

TABLE 4-2
COMPARISON OF RADIX AND RADIX MINUS 1 COMPLEMENTS

Number	Radix Complement	Radix—1 Complement
5	00005	00005
4	00004	00004
3	00003	00003
2	00002	00002
1	00001	00001
+0	00000	00000
-0	undefined	77777
-1	77777	77776
-2	77776	77775
-3	77775	77774
-4	77774	77773
-5	77773	77772

b. A positive number is indicated by the presence of a zero in the most significant digit's position.

c. The transition through zero is smooth; i.e., 00002; 00001; 00000; 77777; 77776.

d. A point of ambiguity exists at 40000 (in this case). This limits the magnitude of positive numbers to 37777 and negative numbers to −37777.

e. An erroneous result or an erroneous indication is obtained when the sum of the factors exceeds the limitations of the machine.

Exercises (261):

1. When you are using complement arithmetic, how many combinations are possible for the solution of a problem?

2. Using radix complements, solve the following and show the answers in octal:

a. $\begin{array}{r} 035_{10} \\ \underline{-023_{10}} \end{array}$ b. $\begin{array}{r} 055_{10} \\ \underline{-036_{10}} \end{array}$

4-3. Radix − 1 Complement Arithmetic

Heretofore, consideration has been given to what is most commonly termed "radix complement arithmetic." In this method, the complement is formed by subtracting the number from a complete power of the base. Thus, it is customarily abbreviated as radix complement. Another form of complement arithmetic, the radix −1 complement, is found to be uniquely convenient in binary arithmetic. The radix −1 complement of a number may be obtained simply by subtracting 1 from the complement obtained by the radix complement method.

262. Solve mathematics problems using the radix minus one method and state selected characteristics of the method.

Radix −1 Operations. Table 4-2 shows a comparison between the numbers +5 and −5.

The salient point here is the transition from positive numbers to negative numbers through zero. Note that when you are using radix −1 complement arithmetic, there exists a positive and a negative zero. Now however, if the largest positive number is permitted to be 37777, the complement of this is 40000. Thus, the previous point of ambiguity no longer exists at 40000. Once again, the absolute value of the largest possible number that the machine can contain is 37777.

Consider some problems using radix −1 complement arithmetic.

(1) (+5) − (+4) = 1

$$\begin{array}{l} 00005 \\ \underline{77773} \\ \boxed{1}\ 00000 \\ \quad \xrightarrow{\qquad} 1 \\ \overline{00001} \end{array}$$

Here an important difference in the two systems is demonstrated. In the radix complement system, an overflow exceeding the limits of the machine had no significance and was dropped; the adder was said to be open-ended. However, in radix −1 complement arithmetic, the overflow is of significance and, as shown in the example, must be added to the least significant digit of what might be termed the partial sum. Thus, a radix −1 complement adder is referred to as having end-around carry.

(2) (−6) + (−6) = −14

$$\begin{array}{l} 77771 \\ \underline{77771} \\ \boxed{1}\ 77762 \\ \quad \xrightarrow{\qquad} 1 \\ \overline{77763} = -14 \end{array}$$

(3) (32000) + (32000) = 64000

$$\begin{array}{l} 32000 \\ 32000 \\ \overline{64000} = -13777 \end{array}$$

This is the same example that was used with the radix complement method. This capacity of the machine has been exceeded, and overflow into the sign yields what appears to be a negative result.

(4) (−37775) + (−37774) = −77771

$$\begin{array}{l} 40002 \\ \underline{40003} \\ \boxed{1}\ 00005 \\ \quad \xrightarrow{\qquad} 1 \\ \overline{} \\ 00006 \end{array}$$

Here is another example of exceeding the negative capacity of the machine. This also produces an erroneous answer.

(5) (+5) − (+5) = 0

$$\begin{array}{l} 00005 \\ \underline{77772} \\ 77777 = -0 \end{array}$$

(6) (+0) + (+0) = +0

$$\begin{array}{l} 00000 \\ \underline{00000} \\ 00000 = +0 \end{array}$$

(7) (−0) + (−0) no equivalent in traditional arithmetic

$$\begin{array}{ll} & 77777 \\ & \underline{77777} \\ 0 & \boxed{1}\ 77776 \\ & \quad \xrightarrow{\qquad} 1 \\ & \overline{77777} = -0 \end{array}$$

(8) (−0) + (+0) no equivalent in traditional arithmetic

$$\begin{array}{l} 77777 \\ \underline{00000} \\ 77777 = -0 \end{array}$$

Examples 5 through 8 demonstrate the conditions resulting in a zero answer. Observe that only in the addition of a positive zero and a positive zero is the sum a positive zero. All other manipulations result in a sum of negative zero.

The salient facts about radix −1 complement arithmetic are:

a. The complement is indicated by the presence of the highest digit of the base being used in the most significant digit's position.

b. A zero in the most significant digit's position indicates a positive number.

c. The transition from positive numbers to negative numbers through zero is not smooth. Zero is represented by 00000, which is termed a positive zero, and by 77777, which is termed a negative zero.

d. Use of either value for zero does not invalidate the result.

e. An erroneous resultant is obtained if the sum of the factors exceeds the limitations of the machine.

f. End-around carry is a necessity when you are using radix −1 complement arithmetic. Overflow into bit position 2^{15} (which does not physically exist in a 15-bit machine, since the bit positions are usually numbered from right to left; i.e., 2^{14}, 2^{13}, . . . 2^{0}) is brought around and added to the least significant digit's position.

A few comments upon the relative merits of the two systems in order. The main advantage of the radix complement system is found in its smooth transition through zero. However, certain difficulties are encountered in designing the circuitry necessary for complementing. While we shall not elaborate on these at this time, you should be aware that generally they limit the use of radix complement arithmetic to countertype applications. In countertype applications, there is no interest in negative resultants; furthermore, all complements are normally entered by the operator or as a fixed function of the machine. The great simplicity in developing the complement in the radix -1 system makes this the most frequently used in the arithmetic section of computers. To demonstrate this, consider the binary equivalent of 05452_8 and its complement in each system.

```
Radix complement:
000 101 100 101 010 = 05452
111 010 011 010 110 = 72326

Radix - 1 complement:
000 101 100 101 010 = 05452
111 010 011 010 110 = 72325
```

Notice that each binary digit is reversed or complemented in the radix -1 system, whereas the reversal in the radix complement system occurs only after the least significant digit. Note also that in forming the radix complement above, the first complemented digit occurs in the 2^2 bit position, and 2^1 is the first bit position containing a 1.

For some applications, the frequent occurrence of a negative zero in radix -1 arithmetic may be objectionable. Several methods have been developed to overcome this limitation. One of these is to detect for the occurrence of the negative zero and, under its stimulus, generate a false carry that will complement the negative zero. Thus:

```
    101 010 111 100 011    =    52743
    010 101 000 011 100    =    25034
    -------------------         -----
    111 111 111 111 111    =    77777
    false carry       1             1
    -------------------         -----
[1] 000 000 000 000 000       [1] 00000
 \                            /
  Overflow             Overflow
```

In this particular instance, the further carry produced by summing with the false carry is disregarded. (This method will not be explored any further at this time, since a true appreciation of the system requires a more complete knowledge of mechanization techniques.)

Another method for circumventing the production of negative zeros is through the use of a subtractive adder (i.e., an arithmetic unit that can only subtract). Just as it is possible to subtract by using a unit capable only of addition, so it is possible to add while limited to the process of subtraction. This is clearly demonstrated through the use of the following equation:

$$a + b = a - (-b)$$

There is only one new concept that you must consider when using a subtractive adder, and that is end-around borrow. This is demonstrated by the following problem:

```
(00047) + (00023) = 00072
[1] 00047
 |  77754
 |  -----
 |  00073
 └─→   -1
    -----
    00072
```

Note that 1 is subtracted from the difference and added in the most significant digit's position of the minuend whenever the subtrahend is greater than the minuend. Although the actual circuitry using this technique does not step through the problem in precisely the manner indicated above, the example effectively represents the action of the machine.

It now remains to consider the results obtained with problems that will produce a zero difference:

```
(1) (00054) - (00054) = 0
            00054
           -00054
           ------
            00000

(2) (-0) - (-0) = 0
            77777
           -77777
           ------
            00000

(3) 0 - (-0) = 0
    [1]     00000
     |      77777
     |      -----
     |      00001
     └───→     -1
            -----
            00000

(4) (-0) - (+0) = -0
            77777
           -00000
           ------
            77777
```

Here note that the only way that a negative zero is obtained is by performing the operation indicated in step 4. Thus, to ensure the normal appearance of a positive zero, it is necessary only to use a subtractive adder. All previously developed rules as applied to radix -1 arithmetic still apply.

Now note these important reminders:

a. The maximum permissible positive value is 37777_8.

b. Addition of two or more factors that exceed this limit will give either an erroneous answer or an answer that is interpreted erroneously.

c. All positive numbers will enter the machine preceded by a most significant bit of zero; all negative numbers will enter the machine in complement form.

d. With the additive adder, a subtract instruction caused the addend to be complemented; in the subtractive adder, an add command will cause the subtrahend to be complemented.

e. Using radix -1 arithmetic, an end-around borrow must be performed whenever the subtrahend is greater than the minuend.

f. The end-around borrow is eliminated by using radix complement arithmetic and a subtractive adder; however, the problem of mechanizing to form the complement is a limitation that precludes the use of this form.

This discussion has proven that the additive process can be made to produce a difference by complementing the addend.

Conversely, the subtractive process can be made to produce a sum by complementing the subtrahend. Either of these concepts may be used in mechanizing the computer.

Using either of these concepts results in a considerable savings, since the same circuits may be used to perform both the add and subtract functions. Either radix or radix -1 complement arithmetic may be used. However the radix -1 complement is easier to mechanize and, hence, is preferred when implementing an adder. The adder will usually be designed as a subtractive adder in order to preclude the frequent occurrence of a negative zero, which is inherent with the use of radix -1 complements in an additive adder.

Exercises (262):
Use radix -1 complement arithmetic and the following problem for exercises 1-3:

$$\begin{array}{r} 067_{10} \\ \underline{+(-010_{10})} \end{array}$$

1. How is the above problem expressed in octal?

2. How is the result contained in a 15-bit register, after end-around carry, expressed in binary?

3. How is the result expressed in octal?

4. What is the main advantage of the radix -1 method in obtaining the complement?

5. What is the main disadvantage of the radix -1 method?

6. Why is a false carry used in some computers using radix -1 complement arithmetic?

4-4. Application

The purpose of the following section is to describe in general terms the processes that may be used by a representative computer performing the arithmetic operations of addition, subtraction, multiplication, and division.

263. State how digit registers operate in the solution of addition and subtraction problems by using radix and radix -1 complement arithmetic, and solve a problem using this arithmetic.

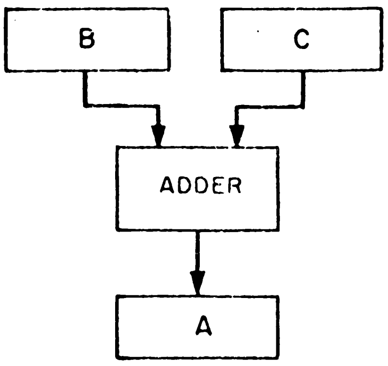


Figure 4-1. Block diagram of add operation.

Addition. Regardless of whether the computer uses an additive or a subtractive adder, it will, during an addition, take the contents of two storage locations or registers and combine them in a third storage location or register in such a fashion as to produce a sum.

The block diagram shown in figure 4-1 illustrates this concept. In the figure, the blocks labeled A, B, and C represent flip-flop storage registers. B and C provide temporary storage for the operands, while A serves as the accumulator. The block labeled "adder" is usually part of the accumulator, serving as its input gating circuitry.

In operation the operands are in some manner gated into the B and C registers. The computer will then execute the instruction:

$$(B) + (C) \longrightarrow A$$

Stated in literal terms, this instruction tells the computer to transfer the sum of the contents of B and C to A. Operations are as follows:

Case 1. Assume that the machine is using the additive process and radix complements.

NOTE 1. In all cases, register size will be limited to 15 bits, bits, with a bit 2^{14} acting as the sign bit LSB = $2^0 1$.
NOTE 2. An erroneous result will be generated should the sum of two positive integers exceed 37777_8 the maximum positive value allowed in the A register.
NOTE 3. Negative values will be entered into the registers in complement form.
NOTE 4. Since binary numbers are cumbersome to handle, their octal equivalents are used to solve the problems.

Problem 1: Solve for the sum of 347 + 405.

$$\begin{array}{rcll} 347_{10} & = & 00533_8 & B \\ +405_{10} & = & 00625_8 & C \\ \hline 752_{10} & = & 01360_8 & A \end{array}$$

Problem 2: Solve for the sum of 347 + (−405).

$$\begin{array}{rcrcrl} 347_{10} & = & 00533_8 & = & 00533_8 & B \\ +(-405_{10} & = & (-00625_8) & = & 77153_8 & C \\ \hline (-58_{10}) & = & (-00072_8) & = & 77706_8 & A \end{array}$$

Problem 3: Solve for the sum of (−347) + 405.

$$\begin{array}{rcrcrl} (-347_{10}) & = & (-00533_8) & = & 777245_8 & B \\ +405_{10} & = & 00625_8 & = & 00625_8 & C \\ \hline +058_{10} & = & 00072_8 & = & \boxed{1}\ 0072_8 & A \end{array}$$

NOTE: The unwanted carry $\boxed{1}$ is dropped.

Problem 4: Solve for the sum of (−347) + (−405).

$$\begin{array}{rcrcrl} (-347_{10}) & = & (-00533_8) & = & 77245_8 & B \\ +(-405_{10}) & = & (-00625_8) & = & 77153_8 & C \\ \hline (-752_{10}) & = & (-01360_8) & = & \boxed{1}76420_8 & A \end{array}$$

Case 2. All conditions are the same except that now radix −1 complements are used.

Problem 1: Solve for the sum of 347 + 405.

$$\begin{array}{rcrl} 347_{10} & = & 00533_8 & B \\ +405_{10} & = & 00625_8 & C \\ \hline 752_{10} & = & 01360_8 & A \end{array}$$

NOTE: When adding positive integers, the operation is the same as when using radix complements.

Problem 2: Solve for the sum 347 + (−405).

$$\begin{array}{rcrcrl} 347_{10} & = & 00533_8 & = & 00533_8 & B \\ +(-405_{10}) & = & (00625_8) & = & 77152_8 & C \\ \hline (-058_{10}) & = & (-00072_8) & = & 777051_8 & A \end{array}$$

Problem 3: Solve for the sum of (−347) + 405.

$$\begin{array}{rcrcrl} (-347_{10}) & = & (-00533_8) & = & 77244_8 & B \\ +405_{10} & = & 00625_8 & = & 00625_8 & C \\ \hline 058_{10} & = & 00072_8 & = & \boxed{1}00071_8 & A \\ & & & & \hookrightarrow 1 & \\ \hline & & & & 00072_8 & \end{array}$$

NOTE: When radix −1 complements are being used, register overflow is treated as an end-around carry. That is, it is added to the LSD (least significant digit) of the partial sum.

Problem 4: Solve for the sum of (−347) + (−405).

$$\begin{array}{rcrcrl} (-347_{10}) & = & (-00533_8) & = & 77244_8 & B \\ +(-405_{10}) & = & (-00625_8) & = & 77152_8 & C \\ \hline (-752_{10}) & = & (-01360_8) & = & \boxed{1}76416_8 & A \\ & & & & \hookrightarrow 1 & \\ \hline & & & & 76417_8 & \end{array}$$

Case 3. Assume that the machine is now using the subtractive process and radix complements. (Notes 1 through 4 still apply.)

Problem 1: Solve for the sum of 347 + 405.

$$\begin{array}{rcrcrl} 347_{10} & = & 00533_8 & = & \boxed{1}00533_8 & B \\ -(-405_{10}) & = & (-00625_8) & = & 77153_8 & C \\ \hline 752_{10} & = & 01360_8 & = & 01360_8 & A \end{array}$$

NOTE: If radix complements are being used and the absolute value of the subtrahend is greater than the absolute value of the minuend, the machine will, in effect, add 10000_8 to the minuend. This process creates a false borrow $\boxed{1}$.

Problem 2: Solve for the sum of 347 + (−405).

$$\begin{array}{rcrl} 347_{10} & = & \boxed{1}00533_8 & B \\ -(+405_{10}) & = & 00625_8 & C \\ \hline (-058_{10}) & = & 77706_8 & A \end{array}$$

Case 4. Assume that the situation is the same as in case 3, with the exception that now radix −1 complements will be used.

Problem 1: Solve the sum of 347 + 405.

$$\begin{array}{rcrcrl} 347_{10} & = & 00533_8 & = & \boxed{1}00533_8 & B \\ -(-405_{10}) & = & (-00625_8) & = & 77152_8 & C \\ \hline 752_{10} & = & 01360_8 & = & 01361_8 & A \\ & & & & \hookrightarrow -1 & \\ \hline & & & & 01360_8 & \end{array}$$

NOTE: As previously stated, when radix −1 complement arithmetic is used, an end-around borrow must be performed whenever the subtrahend is greater than the minuend.

Problem 2: Solve for the sum of 347 + (−405).

$$\begin{array}{rcrl} 347_{10} & = & \boxed{1}00533_8 & B \\ -(+405_{10}) & = & 00625_8 & C \\ \hline (-058_{10}) & = & 77706_8 & A \\ & & \hookrightarrow -1 & \\ \hline & & 77705_8 & \end{array}$$

Problem 3: Solve for the sum of (−347) + 405.

$$\begin{array}{rcrcrl} (-347_{10}) & = & (-00533_8) & = & 77244_8 & \text{B} \\ -(-405_{10}) & = & (-00625_8) & = & 77152_8 & \text{C} \\ \hline 058_{10} & = & 00072_8 & = & 00072_8 & \text{A} \end{array}$$

Problem 4: Solve for the sum of $(-347) + (-405)$.

$$\begin{array}{rcrcrl} (-347_{10}) & = & (-00533_8) & = & 77244_8 & \text{B} \\ -(+405_{10}) & = & (-00625_8) & = & 00625_8 & \text{C} \\ \hline (-752_{10}) & = & (-01360_8) & = & 76417_8 & \text{A} \end{array}$$

Subtraction. The same circuits may be used to perform both addition and subtraction. It requires only a slight change in the operating process.

The circuit shown in figure 4-1 may be used to illustrate this concept. Again, the operands are entered into the B and C registers. Now, however, instead of executing the add instruction, the machine will execute the subtract instruction; that is, it will execute the instruction:

$$(B) - (C) \longrightarrow A$$

This instruction, quite literally, tells the machine to transfer the difference between the contents of the B and C registers to A.

In a machine using an additive adder, this instruction caused the machine to complement the operand representing the subtrahend prior to proceeding with the addition.

Conversely, in a machine using a subtractive adder, the operand representing the subtrahend is not complemented prior to proceeding with the subtraction.

Both of these processes have been demonstrated in the foregoing section on addition.

Exercises (263):

1. Explain how a digital computer uses registers during addition.

2. Into what register is the sum transferred?

3. In a computer using 15-bit registers, what is the maximum positive sum allowed?

4. All negative numbers must be ________ before being entered into the A or B register.

5. When an open-end register is used, the ______ complement method is being used.

6 Solve the following problem, using radix -1 complement arithmetic (leave answer in base 8):

$$\begin{array}{r} 251_{10} \\ -(+370_{10}) \\ \hline \end{array}$$

7. In solving the above problem, what process must the computer first perform?

8. The above problem was arranged for a machine using which operative process?

9. In the above problem, how would the subtrahend be entered?

264. Differentiate among various operations of digital computers during the multiplication process.

Multiplication. Like all other arithmetic operations, multiplication can be accomplished in computers in several ways. One of the most commonly used methods is multiplication by accumulation.

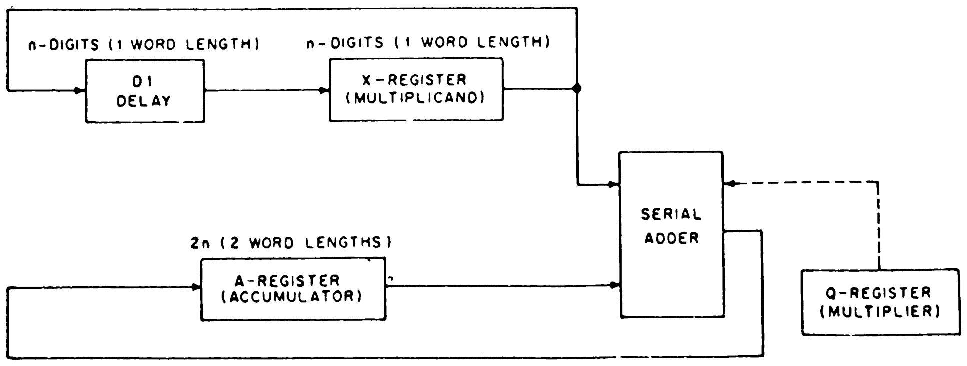


Figure 4-2. Serial multiplication by repeated addition.

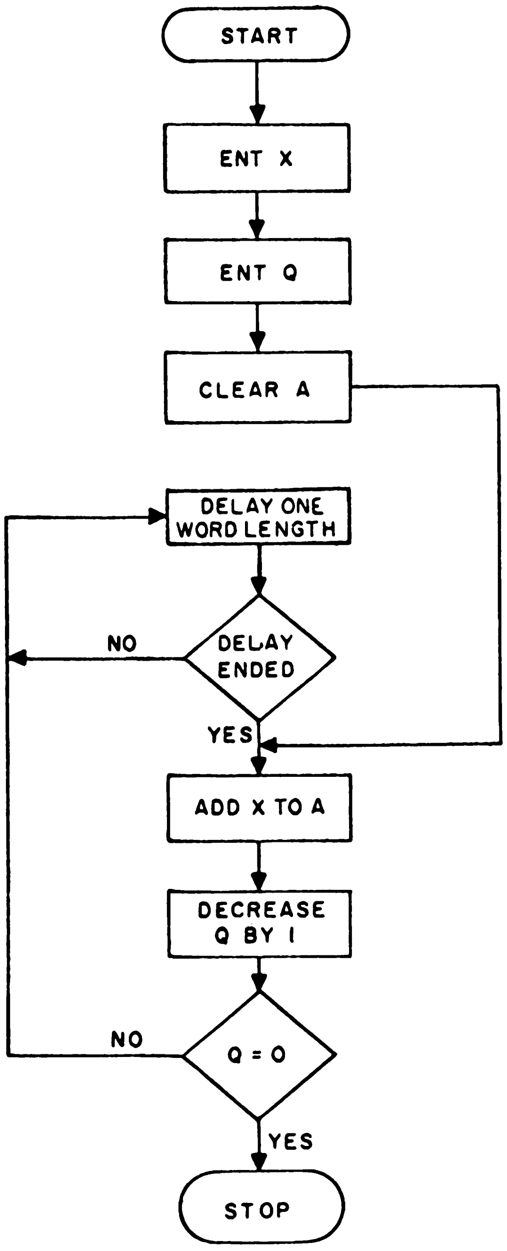


Figure 4-3. Flowchart for serial multiplication by repeated addition.

Serial multiplication by repeated addition. The arithmetic unit of a computer will contain certain registers usually referred to as the X-, Q-, D-, and A- registers. These registers are capable of storing or holding a computer word of a length determined by the design of the machine. In most cases during multiplication, the Q-register, as shown in figure 4-2, holds the multiplier; the X-register, the multiplicand; and the A-register (the accumulator), the sum or partial product. Because the product of two numbers, each containing n digits, can contain as many as 2n digits, the accumulator must be capable of holding twice as many digits as either the multiplier or the multiplicand.

The X-register reads its input (the multiplicand) into the serial adder in serial form at the same time that the accumulator input is being read through in serial form. Initially, the accumulator is cleared to 0.

The multiplier, or Q-register, is actually a counter that determines the number of times the particular multiplicand must bc added to yield the correct product. If the multiplicand is to be multiplied by 3, the Q-register will initially be set to 3 and will count backwards, one digit at a time, as the multiplicand and the accumulator contents are added in the proper order at each count.

Because the accumulator is initially set to 0000 . . . . , the first addition will produce the multiplicand in the accumulator. The second addition is delayed for one word length by D1 to compensate for the difference in word lengths between the accumulator and the X-register inputs. The delay ensures that at the beginning of each count the least significant digits of X- and A-registers will arrive in the serial adder simultaneously.

After the first addition, the contents of the Q-register are decreased by 1, and, after delay, the second addition begins by again adding the contents of the X-register to the accumulated sum. A second delay now takes place in D1 again to ensure that the lowest orders of the X- and A-registers will enter the serial adder on the third count at the same time. The process (diagrammed in the flowchart shown in fig. 4-3) continues until the contents of the Q-register are reduced to 0, at which time the addition has been repeated the number of times indicated by the multiplier. The accumulator contains the product.

Multipliction by using parallel adders. A method of multiplication that uses parallel adders and right shifting is illustrated in figure 4-4. Before proceeding with this method. first observe the example below. Note that the same product is obtained when the addition of the multiplicand is accomplished by using the digits in the multiplier either from right to left or from left to right.

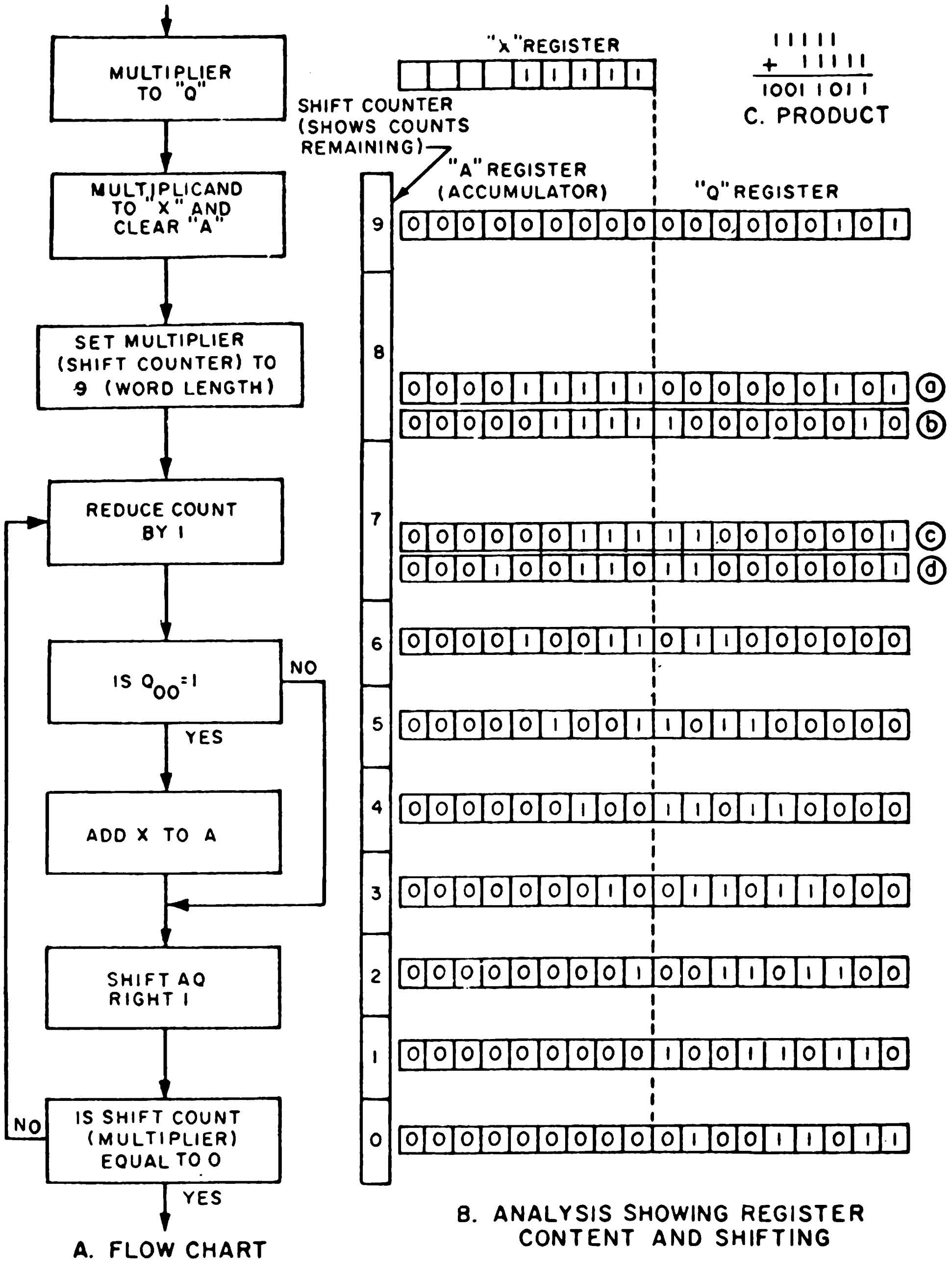
Problem:

```
                 234  Multiplicand
                ×123  Multiplier

  234                      234
   234                     234
   234                     234
    234                   234
    234                   234
    234                  234
  -----                  -----
  28782                  28782
  Equivalent of          Equivalent of
  left-to-right shift    right-to-left shift
```

Thus, multiplication can be (and is) performed by either left-shifting or right-shifting the multiplier.

The flowchart in figure 4-4,A, presents the sequence of events necessary to arrive at a desired result (in this case, the product). All of the steps involved in arriving at this product are accomplished under the direct influence of the control unit. Assume that the multiplicand is 11111 (31) and that the multiplier is 101 (5) in the following example. The control sequences are as follows:



MULTIPLIER TO "Q"
MULTIPLICAND TO "X" AND CLEAR "A"
SET MULTIPLIER (SHIFT COUNTER) TO 9 (WORD LENGTH)
REDUCE COUNT BY 1
IS Q_{00}=1
NO
YES
ADD X TO A
SHIFT AQ RIGHT 1
IS SHIFT COUNT (MULTIPLIER) EQUAL TO 0
NO
YES
A. FLOW CHART
"X" REGISTER
SHIFT COUNTER (SHOWS COUNTS REMAINING)
"A" REGISTER (ACCUMULATOR)
"Q" REGISTER
11111
+ 11111
1001 1011
C. PRODUCT
a
b
c
d
B. ANALYSIS SHOWING REGISTER CONTENT AND SHIFTING
34-1652

Figure 4-4. Multiplication by addition and shifting.

a. Transfer the multiplier (101) to the Q-register (see fig. 4-4,B).

b. Place the multiplicand in the X-register and clear the accumulator (A-register). Set the shift counter to equal the word length (9 bits in this case). The next command is to reduce the shift count by 1 (this does not produce a shift in the AQ-register at this time). Now examine the lowest order bit in the Q-register. If this bit is a 1 (i.e., if Q_{00} = 1) add X (multiplicand) to the A-register. This action places 11111 in the accumulator. The next command is to shift the contents of the A- and Q-registers to the right one bit. In executing this function, the A and Q registers are joined together as a single 18-bit register and the least significant digit in the A-register is shifted into the higher order bit position in the Q-register. This action drops the 1 bit in the lowest order of the Q-register, and the 0 in the second order is now the least significant digit in the Q-register.

c. At this point, you examine the shift counter, to determine whether the count has been reduced to 0. If the answer is "no," as it will be in this case (since eight counts remain), the command is issued to reduce the count by 1, and a subsequent command examines the Q-register to determine whether the lowest order bit is a 1. In this case, the answer will be "no," since the 0 bit has been shifted to the least significant bit position. Because this answer is "no," it is not necessary or required that the "add X to A command" be

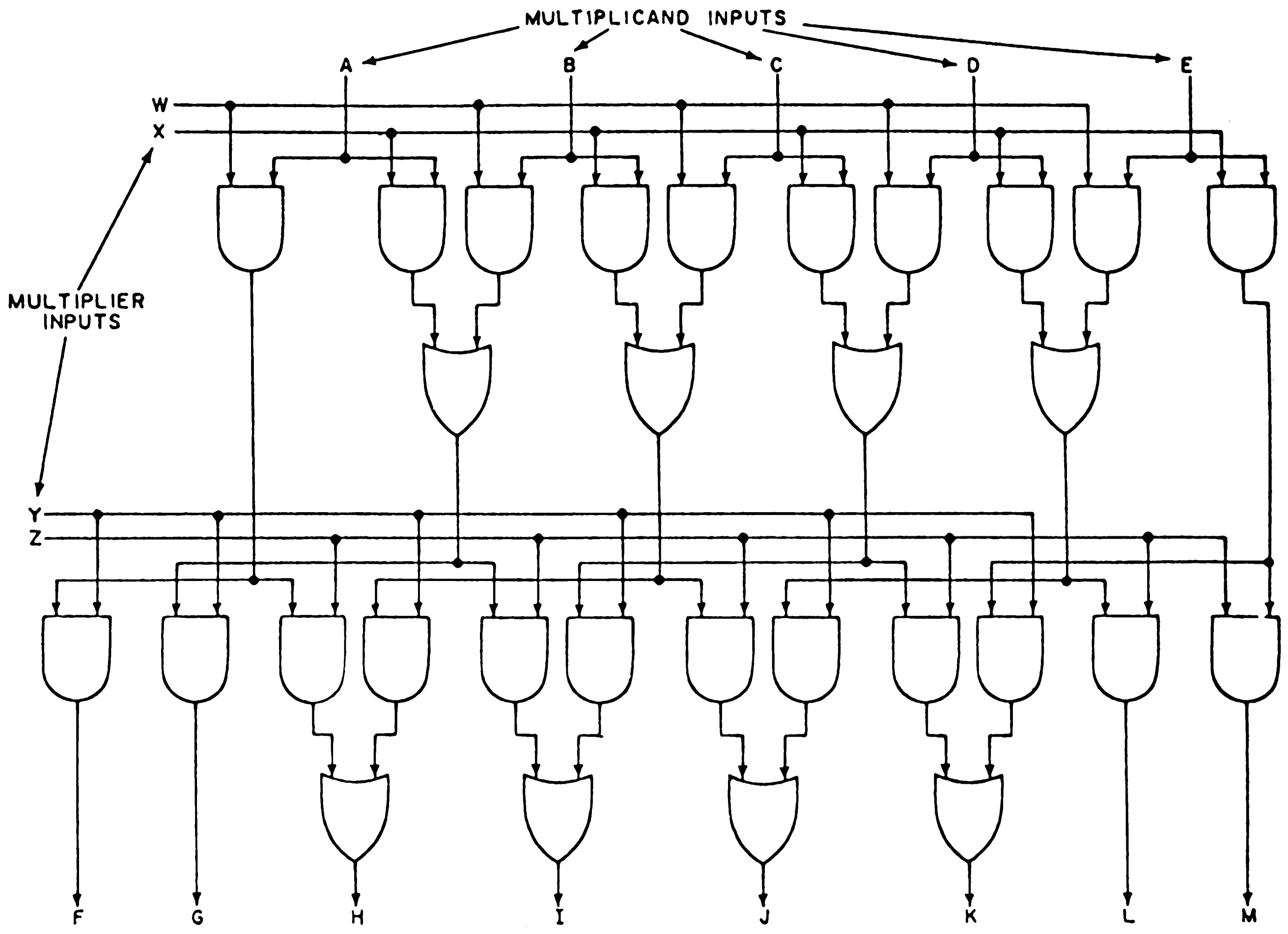


Figure 4-5. Parallel left shift logic circuit.

issued, and this step is bypassed, as indicated in the flowchart. The next command, therefore, shifts all bits in the A- and Q-registers to the right one-bit position, and a 1 bit now appears as the least significant digit in the Q-register.

d. The following step again checks the shift counter to see whether the multiplier count has been reduced to 0. The answer is again "no," as there are seven additional counts. Thus, the command to reduce the count by 1 is again issued. Upon checking the Q-register, the lowest order bit is 1, and the command will be issued to add X (multiplicand) to A. This action adds the multiplicand to the previous accumulator sum as described in the following paragraph.

A check of the shift counter reveals that the count has not been reduced to 0 and that six counts still remain. Because each of these counts produces a "no" answer when the Q-register is checked to determine whether the lowest order bit is 1, the shift AQ right one command will be issued in sequential order as each bit is checked until the shift counter shows the multiplier count to be reduced to 0. At this time, the lowest order bit in the product will be in the lowest order position of the Q-register, followed at the left by each higher order bit, revealing the highest order bit in the product in the 8th bit position to the left. Thus, the Q-register now contains the true product.

Multiplication can also be accomplished by adding and shifting. Either left or right shifting may be used. One circuit for performing a left shift is illustrated in figure 4-5.

This method basically involves left-shifting of the multiplier (zero, one, two, or three positions in this case) and adding whenever a 1 bit is encountered in any one of the multiplier orders.

The number of switching elements used is determined by the number of digits to be multiplied. The multiplicand is applied on lines A through E and remains as the static input throughout the multiplication process. The multiplier digits are applied as control inputs on lines W, X, Y, and Z.

If a 1 bit is present in the first order of the multiplier input lines (multiplication control lines) X and Z are placed in the 1 state. A careful study of the AND and OR elements will reveal that a nonshifted output is developed at the lower terminals I through M. This output represents a partial produce, which is fed to the adder circuits in the accumulator. Now, if the second order digit of the multiplier is a 1 bit, a shift of one place to the left is desired before addition and lines W and Z are in the 1 state. Thus, a left shift is produced (one place to the left) in the upper part of the circuit and is passed through the lower section under the influence of the Z input. The output from terminals H through L is added in the accumulator to form a second partial product.

If the third digit of the multiplier is a 1, the multiplication control lines, X and Y, are placed in the 1 state. This produces a third partial product output between terminals G and K, which is fed to the adder circuits in the accumulator.

Similarly, the multiplication control lines W and Y are used to generate the fourth partial product output between terminals F and J if the fourth place digit of the multiplier is a 1. The output is added in the accumulator to produce the true product.

This particular circuit can also be used to execute multiple shifts in one operation, if up to three of the lower order bits of a multiplier are all zero. This action circumvents the time-consuming process of performing single shifts. The multiple shift can be used, since the accumulator content will not be changed when the multiplier bit is zero.

A sensing circuit (now shown) is used to determine the number of zeroes in the multiplier (up to three), counting from the least significant bit position. This sensing circuit must then enable the proper multiplier input lines (as previously discussed) in order to execute the desired shift.

Exercises (264):

1. Explain the operation of the Q-register when the serial multiplication by repeated addition is used.

2. Tell how the multiplier is manipulated during the parallel adder method of multiplication.

3. What is the purpose of the delay circuit in computers using serial multiplication by repeated addition?

4. In using parallel adders for multiplication, what is the first step?

5. In using the parallel adder method, which registers are combined?

6. In the parallel adder described in the text, where is the final product contained?

7. Refer to figure 4-5. An input of 1 on both X and Z lines will cause an output on lines ________ through ________.

8. Refer to figure 4-5. When the control lines X and Y are selected, in which place is the multiplier a 1?

265. Name methods and cite operating principles used by digital computers in the division process.

Division By Subtraction. Division can be accomplished by repetitive subtraction, as illustrated in the following example using 36 as the dividend and 12 as the divisor. The dividend is reduced by the amount of the divisor for each subtraction.

$12\overline{)36_{10}}$

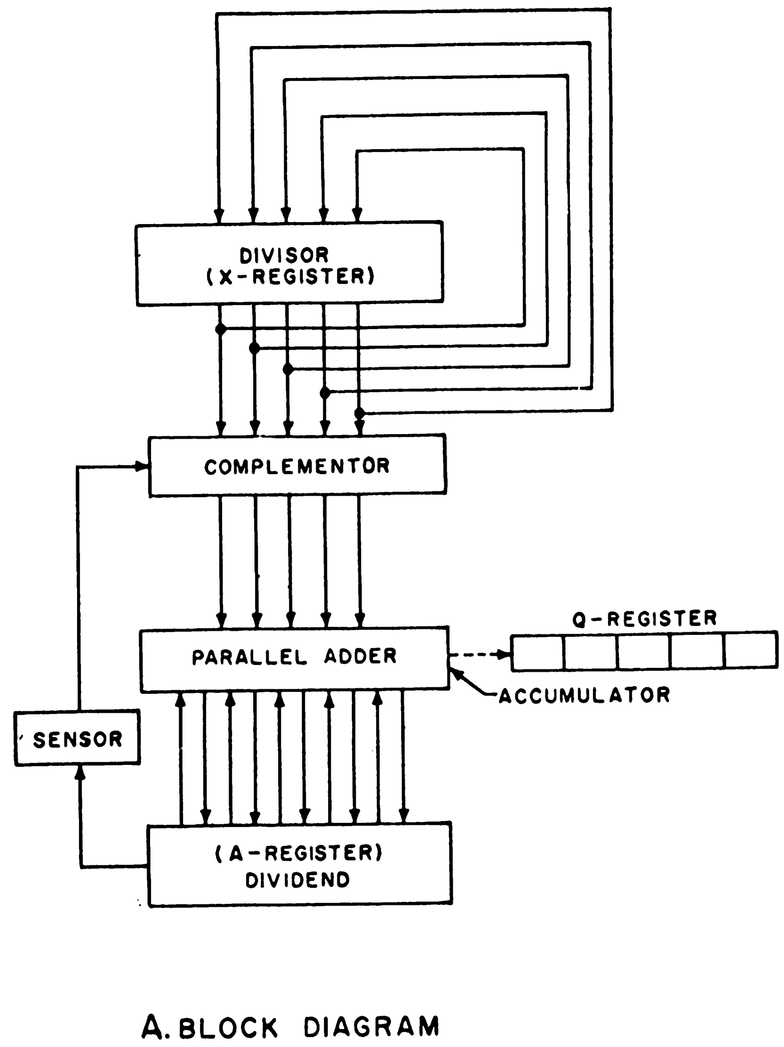
```
 36
-12  first subtraction
 24
-12  second subtraction
 12
-12  third subtraction
  0
```

The number of subtractions completed is the quotient, in this case, three. In binary form, the same example would be:

$$\begin{array}{rl}
100100 & = 44_8 \\
-1100 & = 14_8 \text{ first subtraction} \\
\hline
11000 & = 30_8 \\
-1100 & = 14_8 \text{ second subtraction} \\
\hline
1100 & = 14_8 \\
-1100 & = 14_8 \text{ third subtraction} \\
\hline
000000 & = 0
\end{array}$$

A block diagram for parallel repetitive subtraction is illustrated in figure 4-6,A. You will recall that binary subtraction can be accomplished by complementing the subtrahend and adding. This is the process used in the circuit of figure 4-6,A. The divisor is stored in the X-register, the dividend is in the accumulator (A-register) and the quotient (that is, the number of times the divisor is taken from the dividend) is stored in the Q-register. Because it is necessary to subtract the divisor from the dividend several times, depending on their relative magnitudes, the X-register output is applied to the complementor and recirculated so that it is again stored in the X-register. This makes possible the read-in of the divisor to the complementor as many times as is necessary to reach the final quotient.

When the repetitive subtraction method is used for division, it is possible to obtain a positive remainder, a negative remainder, or 0. When a positive remainder is obtained, the subtraction at that level is valid and a bit is entered into the Q-register. However, when a negative remainder is obtained, it implies that the divisor is larger than the dividend and that the subtraction process has been carried one step beyond that necessary to obtain the final integral quotient. When a 0 is obtained, the divisor will also be subtracted from the 0 and a negative remainder will again be obtained. This too, implies that the subtraction has proceeded one step too far. Thus, it is necessary to use a sensor circuit that is capable of determining whether the remainder is positive or negative. If the remainder is positive, the sensor must allow the divisor input from X to enter the complementor so that the divisor can be complemented and added. The sensor element must permit complementing each time the remainder is positive.



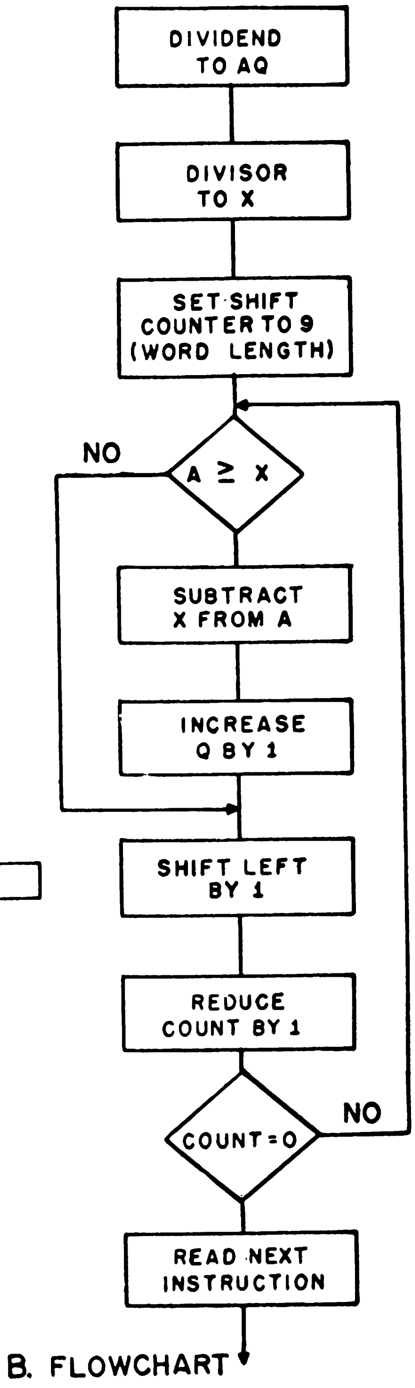


Figure 4-6. Parallel repetitive subtraction.

When a negative remainder is obtained (as a result of a 0 difference or a divisor that is larger than the dividend), the sensor must feed the correct voltage to the complementor to prevent complementing so that on the next cycle the amount subtracted on the previous step will be added back in. Thus, the step beyond that necessary to obtain the integral quotient is nullified.

Although this is a rather simple procedure, it is not very practical when we consider the amount of time necessary to divide a large dividend by a small divisor.

Long-Hand Division. A more practical method involves division by the familiar long-hand method. In this method, you make an inspection to determine the number of times that the divisor can be subtracted from the highest order quantity that is greater than the divisor within the dividend, entering this number as the first digit in the quotient. You then get the remainder and shift the divisor one position to the right. Here you determine how many times the divisor can be subtracted from this portion of the dividend, and you enter this number as the second highest order digit in the quotient. After subtraction to obtain the remainder, you again shift the divisor to the right one position, repeating the process until the remainder after the subtraction from the lowest position is less than the divisor. This process yields the final quotient. This process is illustrated below, using decimal numbers, as follows:

```
     3117.
  15|46763.0
     45
      17
      15
       26
       15
       113
       105
         8
```

Binary Division. In binary division (and common to the shifting method of division), some procedures must be used to determine the number of higher order bit positions of the dividend into which the divisor can first be entered to yield a 1 bit in the quotient. In some computers, an attempt is made to subtract the divisor from the higher orders of the dividend. If the result is positive, the dividend is larger than the divisor, and a 1 bit is entered in the quotient. If the result is negative, the machine automatically negates this step and the divisor is shifted one bit position to the right. This action, in effect, halves the divisor, and, because the dividend now contains one bit position more than the divisor, the subtraction is valid and the dividend proceeds. This procedure is illustrated below.

NOTE: Check for the position of the first bit in the quotient. When the remainder is negative (i.e., not a real number), this step is nullified. Thus:

```
                                       1011(11)
                           1100(12)|10000100(132)
                                    1100
Not a real number                   -100
Nullified                           10000100
Results of right shift               1100
                                      1001
                                      1100
Not a real number                     -101
Nullified                              100100
Results of right shift                 1100
                                        1100
                                        1100
                                        0000
```

Unless the relative magnitudes of the dividend and divisor are restricted to values between two extremes, it is possible that the first subtraction at the higher order can produce a 1 bit and a remainder that is greater than divisor. An example is shown below:

```
        1
  011|11100
      011
      100
```

This inevitably will produce an erroneous result. The error can be prevented by restricting the minimum value of the divisor, the maximum value of the dividend, or both. In the last example, the divisor could be restricted to 100, and/or the dividend 101. In this case, the relative magnitude of the two should be such that the remainder will be less than the divisor after the subtraction of the divisor from the highest orders of the dividend.

In some computers (fractional computers), the quotient is always less than 1 and all significant digits appear to the right of the radix point. In these computers, the above and stated problem is solved by ensuring that the divisor is always larger than the dividend.

The division process illustrated in the flowchart of figure 4-6,B, is similar to that discussed earlier for multiplication except that left-shifting is used instead of right-shifting and a subtraction is the arithmetic operation rather than addition. The AQ-register holds the dividend, and the X-register holds the divisor. The shift-counter contains the maximum number of shifts that may be required (one word length).

The contents of the AQ-register are shifted left one bit position after each subtraction. The highest order bit in the A-register is dropped. At the completion of the division process, the quotient is contained in the Q-register.

Note the comparisons after each left-shift of the AQ contents to determine whether A (the dividend) is greater than or equal to X (the divisor). If $A > X$, then X is subtracted from A. This step is followed by setting Q_{00} to 1; that is, setting the lowest bit in the Q-register to 1. If the $A > X$ comparison results in a "no," a subtraction does not take place.

At the end of the subtraction process (as indicated when the shift counter reaches 0), a "yes" result is produced for the interrogation "is count = 0."

Exercises (265):

1. Name the simplest method of division in a digital computer, and state the major disadvantage of this method.

2. Name a more practical method of division in a computer.

3. In figure 4-6,B, how are the contents of the AQ-register shifted after each subtraction?

4. Again, refer to figure 4-6,B: In which register is the highest order bit dropped? At the completion of the division process, in which register is the quotient contained?

5. In binary division, procedures must be used to determine the number of higher order ________ positions of the dividend into which the divisor can first be entered to yield a ________ ________ in the quotient.

CHAPTER 5

Waveshaping Circuits

THE FUNCTION OF many electronic circuits is waveshaping for timing and control. Waveshaping circuits such as integrators, differentiators, limiters, clampers, and Schmitt triggers produce a variety of waveforms. The duration and/or amplitude of these waveforms are controlled with respect to time by these circuits. Proper operation of a waveshaping circuit will depend on the circuit's response to the transient voltage or current applied.

A transient voltage is an instantaneous surge of voltage which occurs as the result of a change from one steady-state condition to another. Transients can be good if they are intended as a trigger for a logic circuit, or they can be harmful if they occur when not desired. Many an integrated circuit or solid-state device has died from undesired transients.

5-1. Differentiation and Integrators

A circuit's response to a transient waveform will determine what form the signal will have at the output of that circuit. Waveshaping circuits must ensure that timing and data signals adhere to the waveform and voltage specified for the circuits that follow. Differentiation and integration are two processes which provide specific waveforms for a desired circuit operation.

266. Specify the characteristics of differentiation and integration circuits.

Differentiators. Differentiating circuits produce an output voltage that is proportional to the rate of change of the input. Note that in the RC differentiator illustrated in figure 5-1, an output voltage occurs only when the square wave rises or falls. An RC differentiator requires a short-time constant (1/10 the input frequency) with the output taken across the resistor. Remember that the time constant (TC) in an RC circuit is equal to the product of R and C. At T0, the input changes rapidly from one steady state to another; that is, the rate of change is minimum, and the output voltage is maximum. From T0 to T1, there is zero rate of change, and the output drops to zero at T1. How fast the output drops to zero depends on the RC time constant for the circuit which determines the charge rate of the capacitor. At T1, there is another sudden change of direction with a maximum rate of change, and the output voltage is again maximum but in the opposite direction. The sharpness of the output spike depends on the shortness of the RC time constant. Differentiators, such as described here, are used as part of the input circuitry of flip-flops where a sharp spike is needed to trigger the circuit.

Integrators. Recall that integration is the process of summing up an infinite number of minute quantities. An integrating circuit produces an output voltage that is essentially the time integral of its input waveform. In an RC integrator circuit, a long-time constant (10 times the input frequency) is used, and the output voltage is taken across the capacitor. Refer to figure 5-2 for this discussion of an integrator circuit. At time T0, the area under the square-wave input is zero. As time progresses from T0 to T1, the same amount of area is added with each increment of time. The output increases in a linear fashion, and it reaches a maximum at time T1. The input voltage during the time integral from T1 to T2 is negative, and the output decreases linearly toward zero volts.

An integrating circuit can be used as an input to a Schmitt trigger circuit where a predetermined threshold voltage must be reached before the circuit fires or conducts. This threshold voltage could be a function of an integrator circuit that is receiving valid radar returns or tape drive sync-pulse inputs.

We should note here that RC coupling circuits are similar to integrator circuits, and this fact could cause confusion. RC coupling circuits are used primarily to transfer waveforms from one circuit to another so that the output closely resembles the input. Like the RC integrator, RC coupling circuits use a long-time constant (10 times the input frequency), but the output is taken across a resistor.

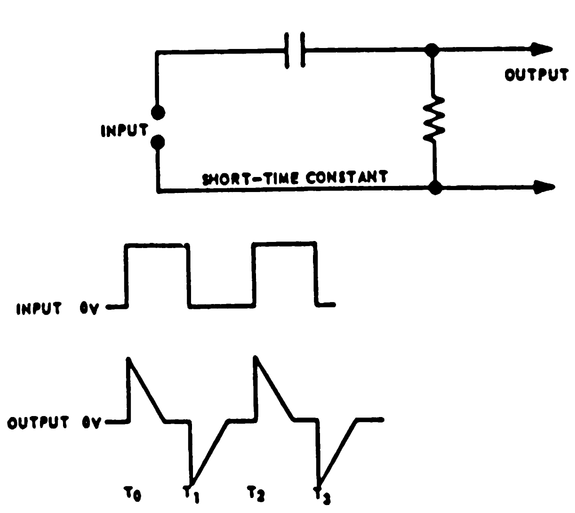
Exercises (266):

1. In the following circuit, C1 is shorted (fig. 5-3). Select the output of the circuit.

2. Why are differentiators used as part of the input circuitry of flip-flops?

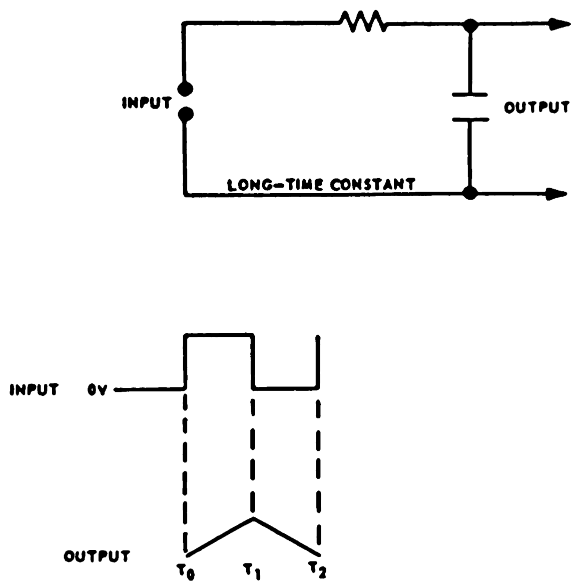
3. List one use for an integrator.

4. Which circuit technique described in this section could also be called a summing circuit?



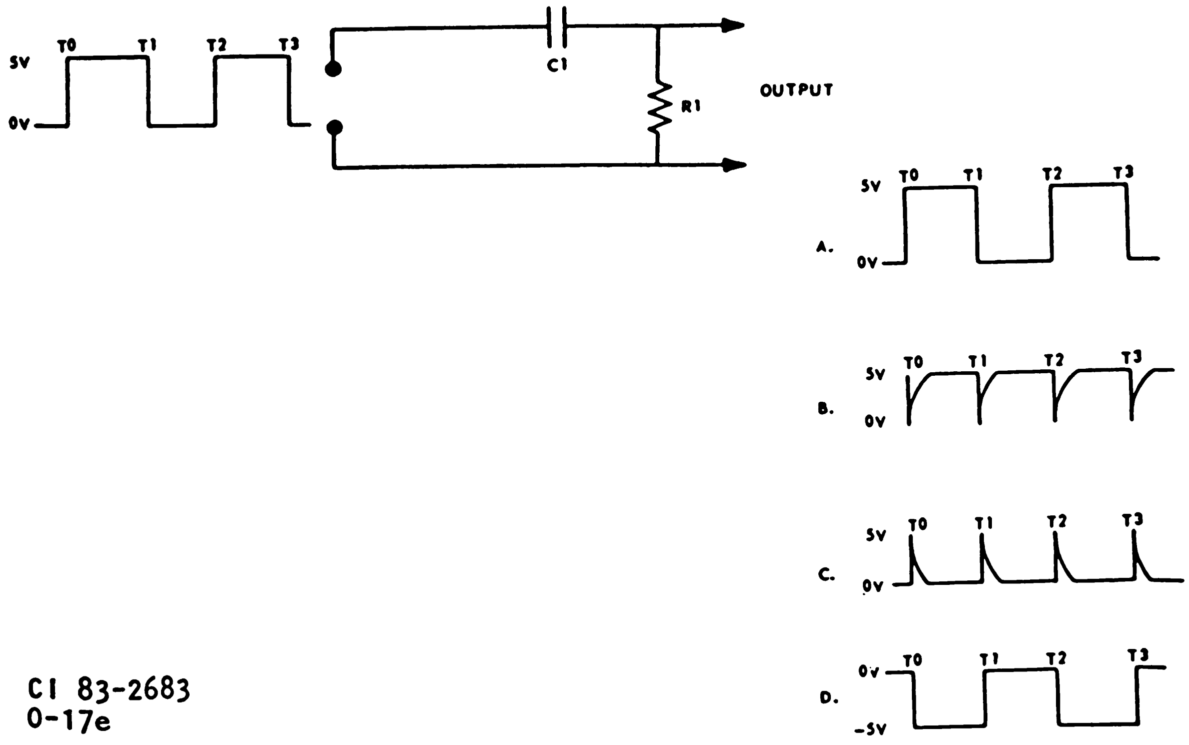
CI 83-2681
0-17e

Figure 5–1. RC differentiator.



CI 83-2682
0-17e

Figure 5–2. RC integrator.



CI 83-2683
0-17e

Figure 5–3. Objective 266, exercise 1.

5-2. Limiters, Clippers, and Clampers

These circuits use active devices such as diodes, tubes, or transistors, to perform the function of controlling signal amplitudes and voltage levels. The use of limiting, clipping, and clamping circuits in electronic equipment ensures reliable operation. Limiters and clippers act to control the amplitude of a signal, and clampers act to establish a relationship between a signal and some desired voltage level.

267. Specify the characteristics of limiters/clippers and clamping circuits.

Limiters. A limiter is defined as a device which prevents some characteristics of a waveform from exceeding a predetermined value. Limiting is used for waveshaping or for circuit protection by preventing a voltage from becoming too large. The two types of limiters discussed here are the series and shunt (parallel).

Series limiter. A limiter can be designed using a diode and a resistor. When the output is in series with the diode, the circuit is called a series limiter. Parts A and B of figure 5-4 are diagrams of a positive and a negative series limiter. Note that the diode allows conduction in only one direction and blocks or limits the signal in the opposite direction. In part A of the figure, the diode is reverse-biased by the positive alternation of the input; this prevents the positive alternation from being developed at the output. The diode is forward-biased by the negative alternation of the input, and this alternation is developed at the output. In part B, the diode is forward-biased by the positive and reverse-biased by the negative alternation. Therefore, the negative alternation is blocked or limited.

Shunt limiter. When the output is in shunt (parallel) with the limiting device, a shunt limiter is formed. Shunt positive and negative limiters are illustrated in figure 5-5, A

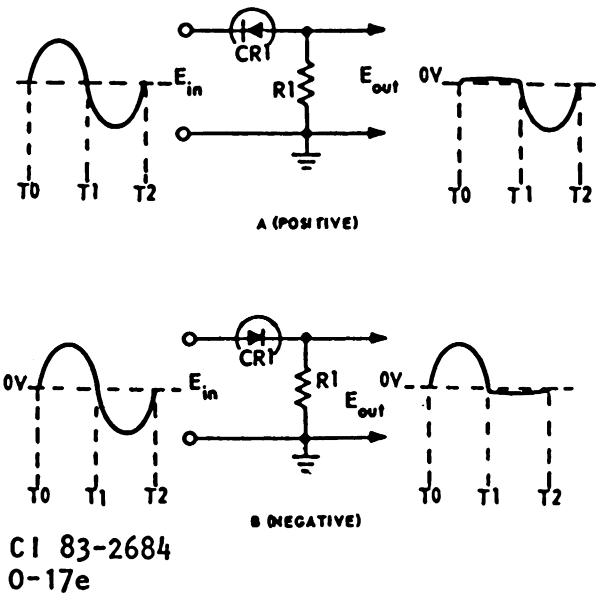


Figure 5-4. Series limiter.

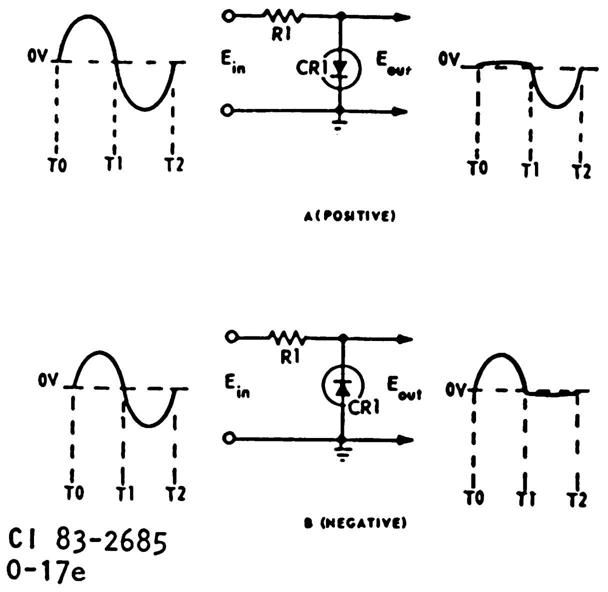


Figure 5-5. Shunt limiter.

and B. The function of the shunt limiter is the same as the series limiter; that is, limits a portion of the input signal. However, note that in the shunt limiter, limiting occurs during the conduction of the limiting device.

In the shunt limiters just discussed, limiting occurred near a zero-reference level. This reference level could be controlled by providing a bias voltage for the limiting device. Depending on the value of this bias voltage, limiting may remove only a portion of one alternation of an input sine wave. Figure 5-6, part A, illustrates a positive shunt limiter with positive bias. Note that in this particular circuit, the battery causes the diode to be reverse-biased until the input goes more positive than a +4 volts. Therefore, limiting does not occur until the input reaches this positive bias voltage level from T_1 to T_2. Note that this is the only time that the diode conducts. Now look at the shunt negative limiter with negative bias, illustrated in figure 5-6, part B. In this circuit, the battery causes the diode to be reverse-biased until the input goes more negative than −4 volts. Once the input reaches this level, the diode conducts—causing limiting to occur from T_2 to T_3. In some texts, limiters are also called clippers. They perform the same limiting (clipping) function discussed here. It is possible, by combining the circuits of figure 5-6, parts A and B, to provide clipping of both the positive and negative peaks of the waveform. Another method would be to use two Zener diodes, rated at 4 volts each, connected back-to-back to limit our waveform to ±4-volt excursions. Whenever we limit a waveform with a limiter, we also reduce the average power available from the signal. This reduction of power may be desired in some applications.

Clampers. There are certain circuit applications within digital electronics which require the upper or lower extremity of a waveform to be fixed at a specific DC level or ground. Clampers are the circuits which perform this function.

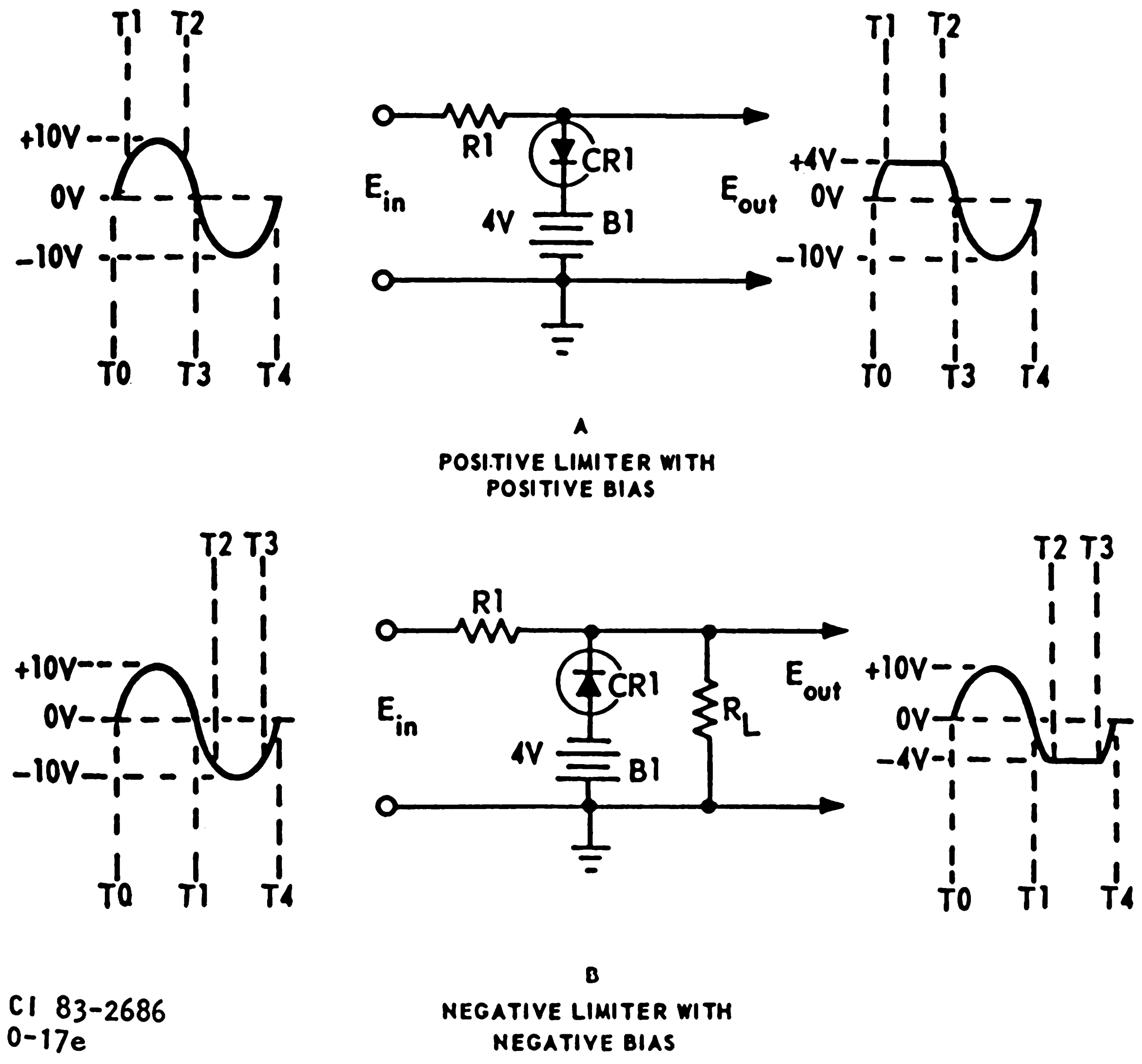


Figure 5–6. Shunt limiter with bias voltage.

Positive clamper. Figure 5-7, part A, illustrates a positive clamper. The identifying feature to note here is that the diode's cathode is connected to the capacitor. At T0, the +25-volt input causes CR1 to conduct and C charges to 25 volts. At T_1, the 25 volts across C and the +25-volt input are series-aiding. Thus, +50 volts appears across R and CR1. At this time, CR1 is reverse-biased. From T_1 to T_2, C discharges to approximately 23 volts (determined by the value of R and C), and the output (part B of the figure) drops from 50 to 48 volts. At T_2, the input is −25 volts, CR1 conducts, and the output goes to approximately -2 volts. From T_2 to T_3, C charges quickly through CR1, from 23 to 25 volts and the output goes from −2 to 0 volt. Note that the peak-to-peak value of the output is the same as the input (50 volts), but its lower extremity is clamped to zero volts or ground. Whenever we clamp a waveform to zero reference or ground, the circuit becomes what is known as a DC restorer.

Negative clamper. The identifying feature of this type of clamper is that the diode's anode is connected to the capacitor. If the clamper in figure 5-7 were a negative clamper, the diode would be turned around (anode connected to capacitor) and the output would vary between 0 and −50 volts.

Establishing the clamping reference. Some circuit applications require a signal clamped to a voltage other than ground. If we add a +10-volt bias battery between the signal-return line and the resistor and diode, as shown in figure 5-8, we would change the clamping reference from 0 to +10 volts. In the positive clamper shown, the minimum value of the signal is clamped to the +10-volt bias. In a negative clamper, the maximum value of the signal would be clamped to this bias.

Determining the output of a clamper. In troubleshooting a clamper circuit, you must know how to determine its output for a given input; otherwise, you will not know whether the output is correct. The following three steps are a guide for determining a clamper's output:

(1) Determine whether it is a positive or negative clamper from the position of the diode in the circuit.

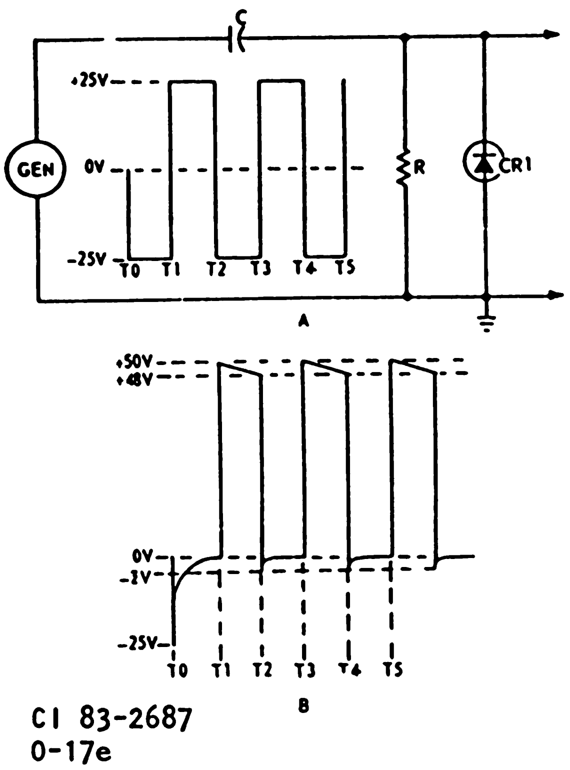


Figure 5–7. Positive clamper circuit and waveforms.

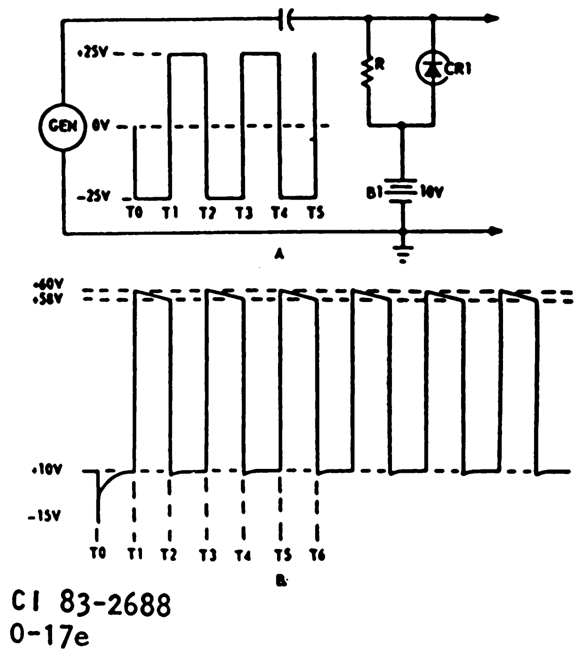


Figure 5–8. Positive clamper with positive bias (10 volts).

(2) Draw the clamping reference level which is ground or a bias voltage.

(3) Draw the input waveshape exactly as it is with respect to shape and peak-to-peak amplitude; however, the lower extremity should be drawn on the clamping reference level for a positive clamper or the upper extremity drawn on the clamping reference level for a negative clamper.

Figure 5-9 illustrates the use of these three steps for a square wave and a sawtooth. Note in part A of the figure that the peak-to-peak amplitude of these waveshapes is 250 volts (−50 to +200 volts). B and C of the figure illustrate positive clamping, because the lower extremity is clamped to the reference level. D and E illustrate negative clamping, because the upper extremity is clamped. This figure also shows that clamping does not change the amplitude of the signal to any great extent.

Exercises (267):

1. In the following circuit (fig. 5-10), the output is incorrect. What is the most probable cause for the incorrect output?

2. Which is the correct output for the following circuit (fig. 5-11)?

3. What is the identifying feature of a negative clamper?

4. What is the clamping circuit used to reference a waveform to a 0 volts or ground?

5-3. The Schmitt Trigger

Digital pulse signals will deteriorate over time, especially when transmitted over long signal lines. The pulses lose their rectangular shape and pick up noise signals. It is then necessary to condition the pulses to restore their original waveshape. The Schmitt trigger circuit of figure 5-12 is designed to perform this function.

268. Specify two purposes for the Schmitt trigger circuit and state how it operates.

Circuit Operation. In its quiescent state, the input to the Schmitt trigger at R1 is at 0 volts and Q1 is cut off. The voltage divider—composed of R3, R4, and R5—divides the source voltages of −12 volts and +12 volts so that the base of Q2 is forward-biased. Q2 will be saturated. The current through Q2 develops a voltage drop across R7 which reverse biases Q1 and keeps it cut off. In this condition, the output taken from the collector of Q2 is nearly 0 volts.

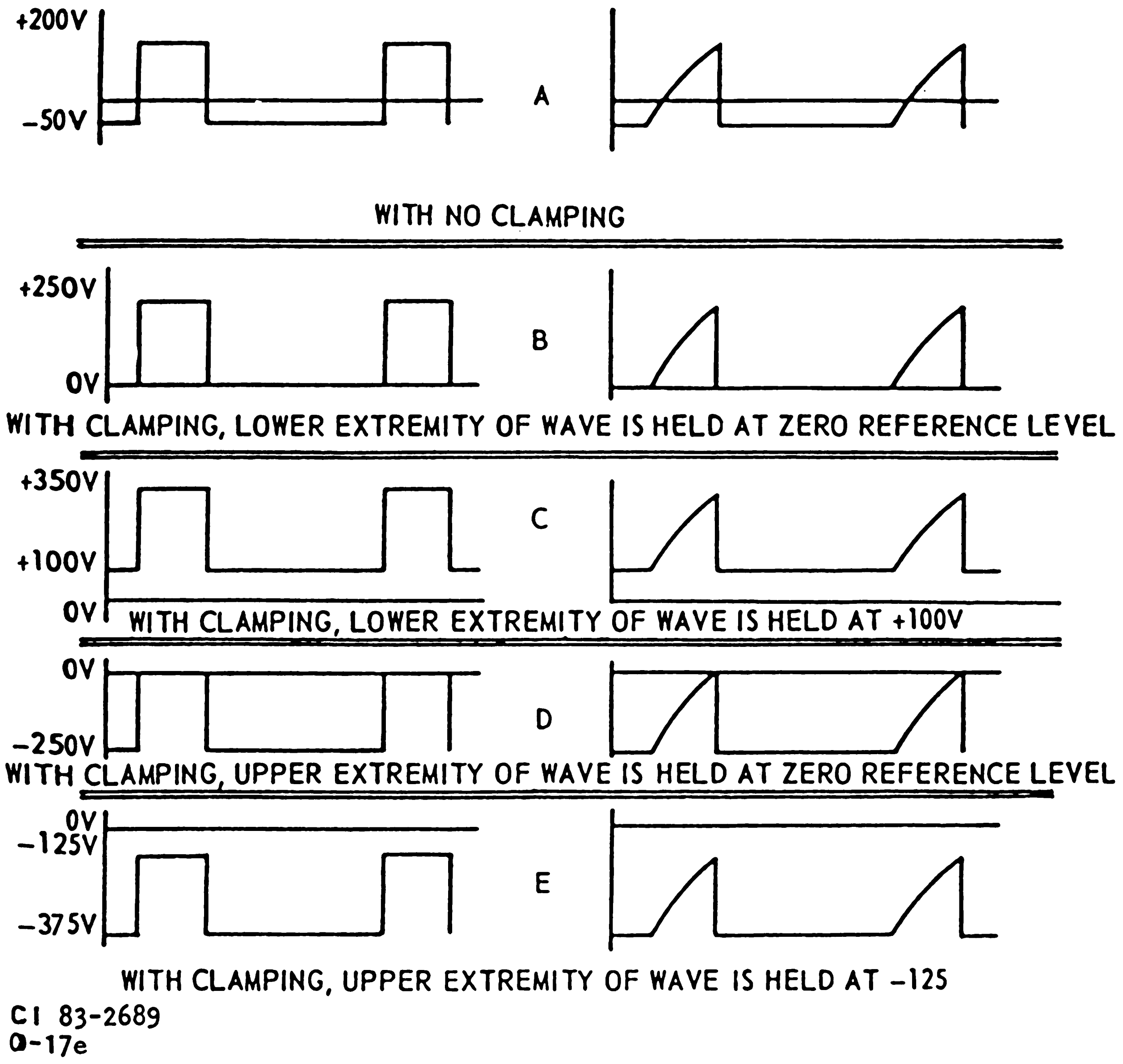

+200V
-50V
A
WITH NO CLAMPING
+250V
0V
B
WITH CLAMPING, LOWER EXTREMITY OF WAVE IS HELD AT ZERO REFERENCE LEVEL
+350V
+100V
0V
C
WITH CLAMPING, LOWER EXTREMITY OF WAVE IS HELD AT +100V
0V
-250V
D
WITH CLAMPING, UPPER EXTREMITY OF WAVE IS HELD AT ZERO REFERENCE LEVEL
0V
-125V
-375V
E
WITH CLAMPING, UPPER EXTREMITY OF WAVE IS HELD AT -125
CI 83-2689
0-17e

Figure 5–9. Clamping waveforms.

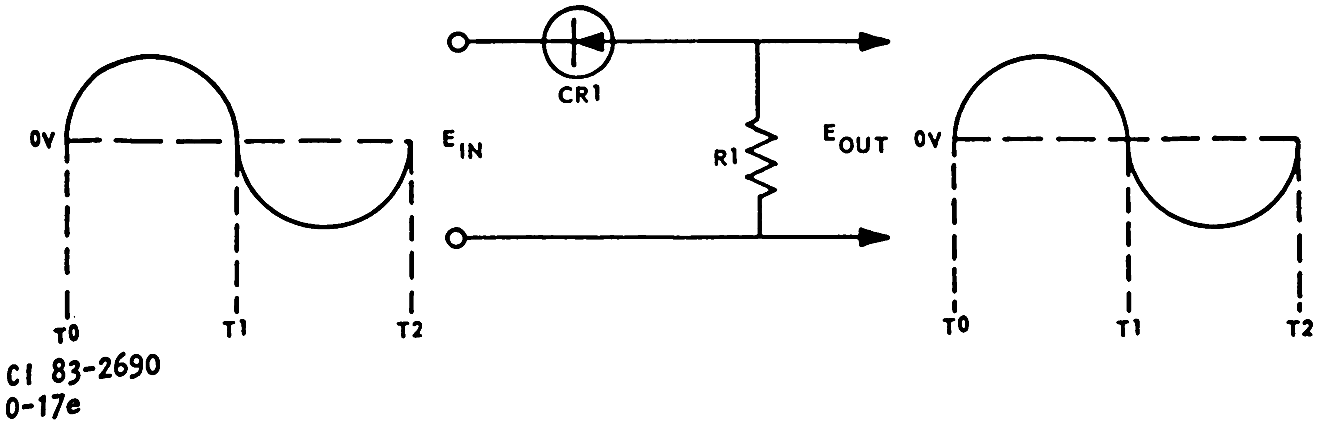


Figure 5–10. Objective 267, exercise 1.

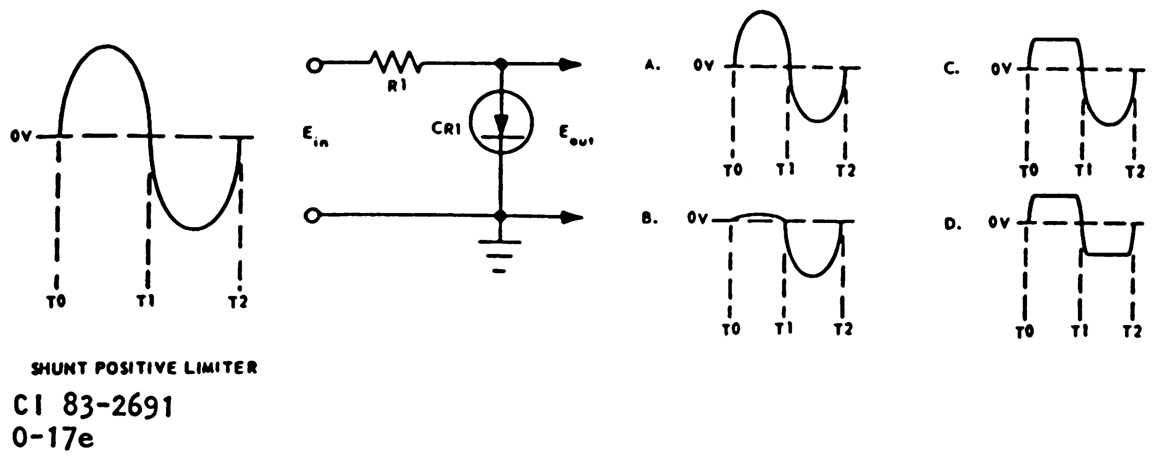


Figure 5–11. Objective 267, exercise 2.

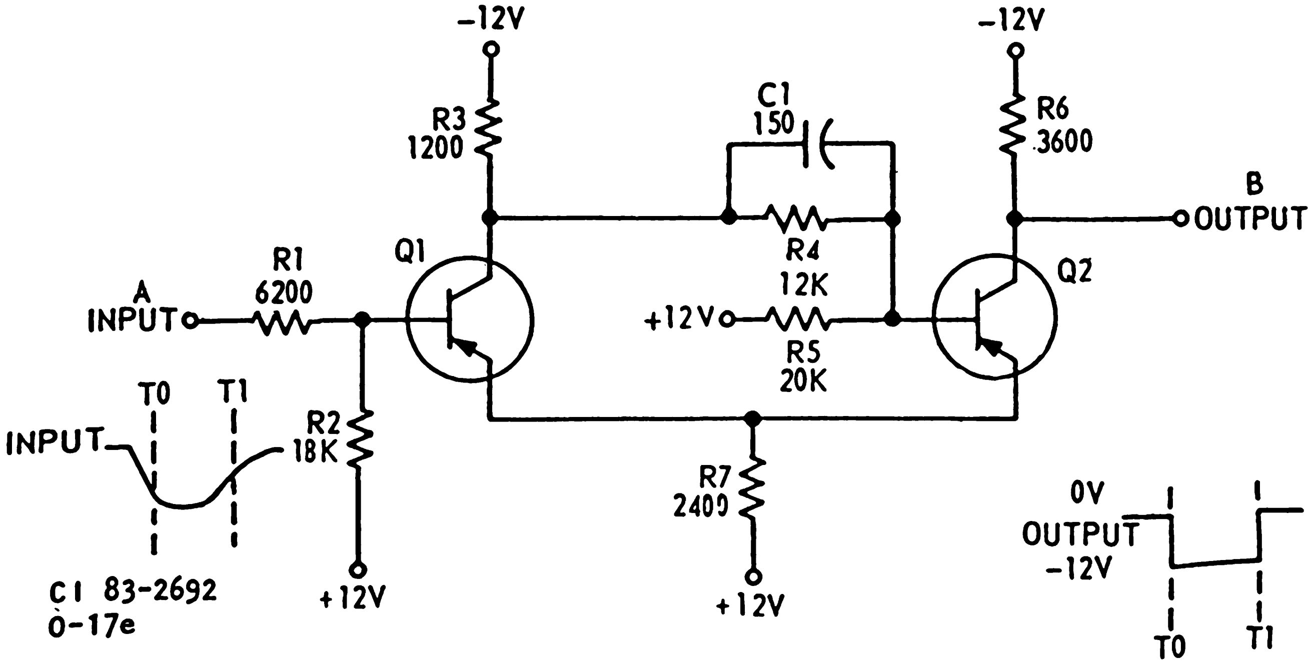


Figure 5–12. Schmitt trigger diagram.

At time T_0, the negative signal applied at A input has sufficient amplitude to bias Q1 on, and its collector goes toward 0 volts. This change is coupled to the base of Q2 and causes Q2 to cut off. The decrease in current through R7 reduces the reverse bias on Q1, causing it to saturate. The collector voltage of Q2 is now -12 volts.

The circuit remains in this state until T_1, when the input voltage becomes less negative, decreasing to a value that causes Q1 to start conducting less. The collector potential of Q1 starts in the negative direction. This change is coupled to the base of Q2 and turns it on. The increase in current through Q2, and R7 puts a reverse bias on Q1 which cuts it off. As a result, Q2 conducts near saturation, and the collector voltage is near 0 volts.

Notice how the rounded input wave is converted to a square-wave output between T_0 and T_1. The sharp rise and fall at the output is due to the feedback between Q2 and Q1. Any slight change in the conduction of Q1 is applied to the base of Q2 which, in turn, changes Q1 emitter voltage. Capacitor C1 speeds the transition from one state to the other.

Applications. Schmitt trigger circuits find applications as squaring and voltage-level sensing circuits. Voltage-sensing circuits are useful in warning or control circuits. If the input voltage rises above or falls below a specified level, the Schmitt trigger produces an output which then actuates warning or correction circuitry.

The Schmitt trigger (ST) function symbols are shown in figure 5-13. The ST is actuated when the input signal crosses a certain THRESHOLD voltage. Output signal amplitude and polarity are determined by the circuit characteristics of the ST and not by the input signal. Waveforms may be shown inside or outside the symbol, indicating amplitude, polarity, and threshold voltage. The unactuated state of an ST is either zero or one. When actuated, it changes to the opposite state and remains in the opposite state as long as the input exceeds the threshold value.

The Schmitt trigger circuit may be used in its discrete form, as discussed here, but it is also available as an integrated circuit. The IC version of the Schmitt trigger circuit works functionally the same as the circuit discussed here.

Exercises (268):

1. What are the two purposes of the Schmitt trigger?

2. Explain what happens when a negative pulse arrives at the input of the Schmitt trigger circuit in figure 5-12.

3. How long will transistor Q1 in figure 5-14 continue to conduct?

5-4. Timing Circuits

Timing circuits are used in computers to ensure the proper timing of events. Timing pulses enable and disable certain circuits, thus permitting certain operations to begin and others to be terminated. The return of these pulses a few microseconds later could disable the circuits, thus terminating those operations previously begun and beginning a new set of operations. Once a resident program has been installed and the computer set in operation, the process of sequentially enabling and disabling circuits continues until one of three things occur—(1) the resident program is completely executed, (2) a programmed stop is reached, or (3) a fault condition occurs.

A pulse generator of a type determined by computer design provides the main timing signal for any given computer. These pulse generators are commonly termed "master clocks," or reference generators, and usually operate by the maximum rate at which the computer handles data.

269. State the difference between single-phase and multiphase systems and how each operates.

Single-Phase Systems. Figure 5-15,A, presents in block diagram a relatively simple, single-phase, masterclock

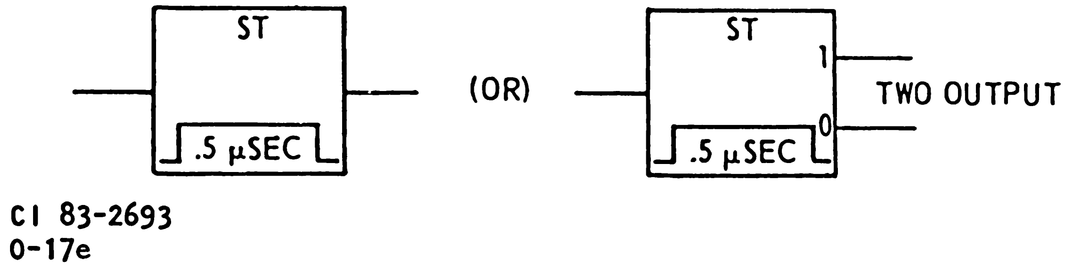


Figure 5-13. Schmitt trigger logic symbols.

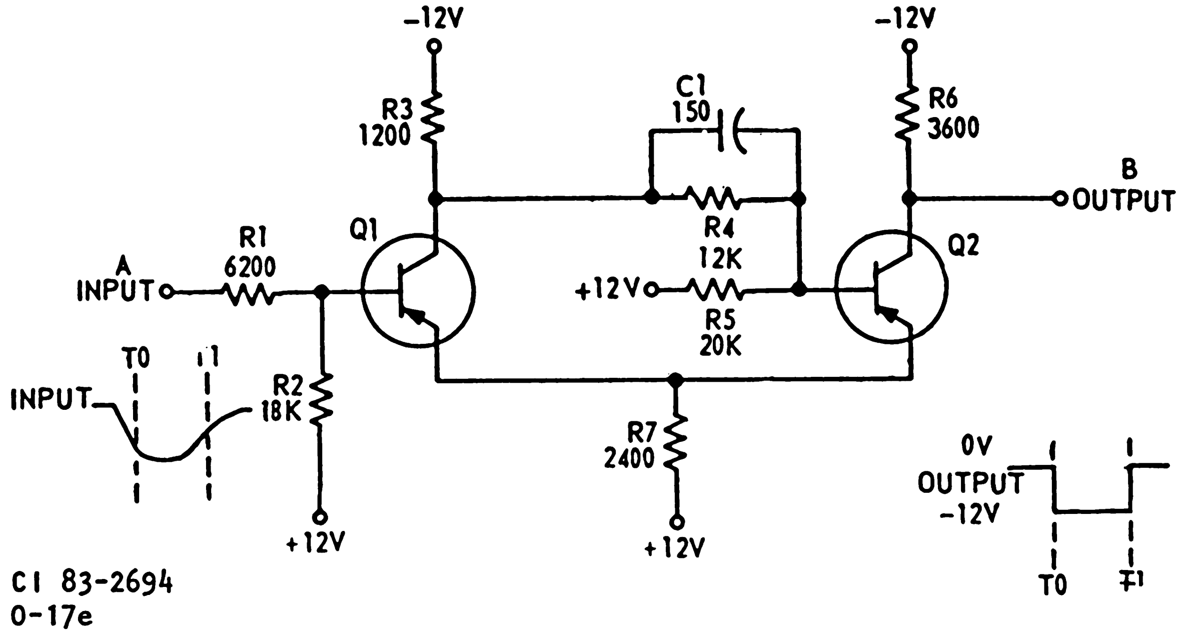


Figure 5–14. Objective 268, exercise 3.

circuit. As indicated, this circuit consists essentially of a free-running (astable) multivibrator and a single-shot (monostable) multivibrator. The astable multivibrator generates the necessary pulses, and the single-shot serves as a pulse shaper. (Fig. 5-15,B, illustrates the relationships existing between the output of the pulse generator and the output of the pulse shaper.) The output of the pulse shaper is then used to enable and disable circuits in whatever sequence is necessary to properly execute the computer program. Single-phase clock systems find limited usage in modern computers and are not described in any further detail.

ART BELOW BAR OVER X________

Multiphase Systems. Consider now the block diagram in figure 5-16,A. In this circuit, both outputs (X and X) of the pulse generator are used to drive single-shots. The output signal of the upper single-shot is designated the odd phase, i.e., $\phi 1$ (CP1). The output signal from the lower single-shot is designated the even phase, i.e., $\phi 2$ (CP0). These two signals are positive-going pulse trains displaced in time from each other by one-half the period of the pulse generator. In other words, they are generated on alternate half cycles of the astable multivibrator signal (see timing diagrams, fig. 5-16,B).

Each of these pulse trains (clock phases) may be used to alternately enable and disable circuits, thus permitting functions involving more than one operation to be

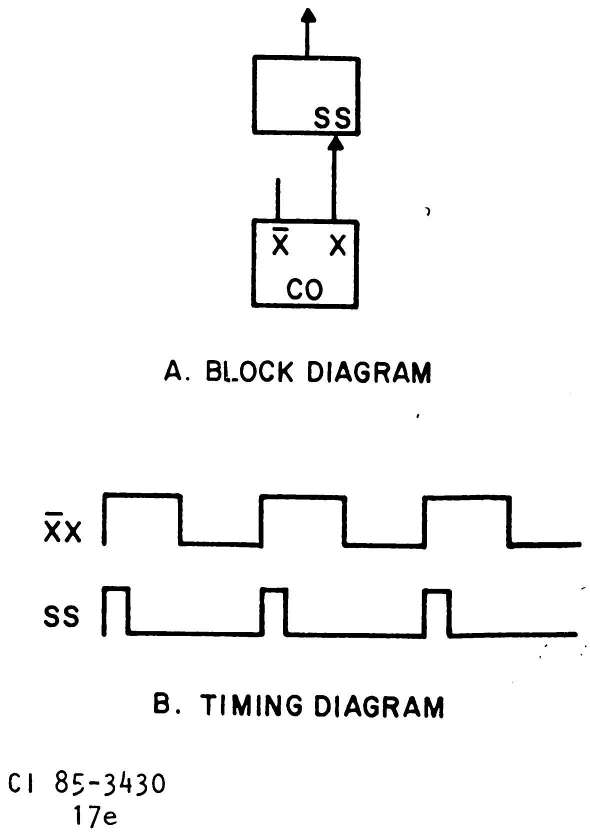
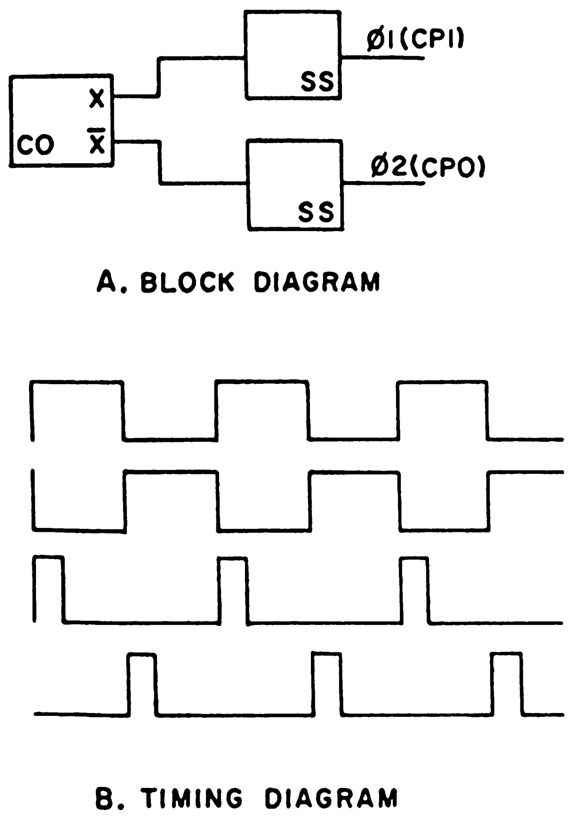


Figure 5–15. Single-phase master clock circuit.



CI 85-3431
17e

Figure 5–16. Two-phase master clock.

completed during a given clock cycle or a given operation to be extended over more than one clock cycle.

When necessary, these two signals may also be used to generate additional phases. These additional phases enable the computer to perform even more complex functions involving multiple operations with varying completion time (some operations may require less than a clock cycle for completion while others may extend over several clock cycles).

Figure 17,A, illustrates one way in which this concept might be implemented. In this circuit, CP1 toggles 2/4 (the even-phase enable flip-flop) and serves as a partial enable to AND gates ϕ1 and ϕ3. CP0 toggles 1/3 (the odd-phase enable flip-flop) and serves as a partial enable to AND gates ϕ2 and ϕ4. (NOTE: When toggled, the enable flip-flops will change states on the trailing edge of the toggle pulse.)

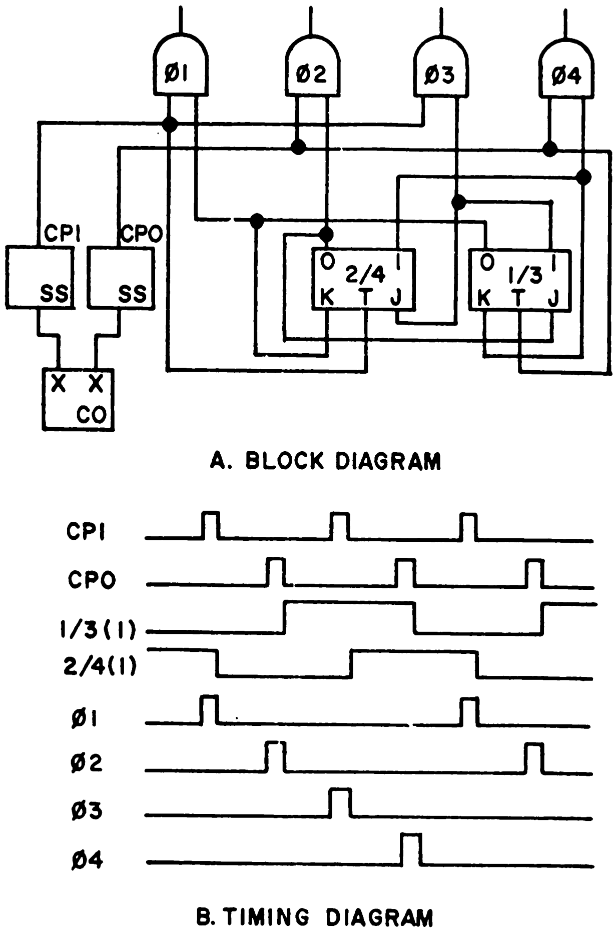
Assuming an initial state where the odd-phase enable flip-flop is in its clear state and the even-phase enable flip-flop is in its set state, circuit operation will be as follows:

(1) When the first CP1 pulse appears, it will, in conjunction with the high from the clear output of the odd-phase enable flip-flop, fully enable AND gate ϕ1, causing it to output a HIGH (a phase 1 clock enable). The first CP1 pulse also toggles the even-phase enable flip-flop to its clear state.

(2) When the first CP0 pulse appears, one-half period later, it, in conjunction with the HIGH from the clear output of the even-phase enable flip-flop, will fully enable AND gate ϕ2, causing it to output a HIGH (a phase 2 clock enable). The first CP0 pulse also toggles the odd-phase enable flip-flop to its set state.

(3) One-half period later, the second CP1 pulse appears, and in conjunction with HIGH from the set output of the odd phase enable flip-flop, it will fully enable AND gate ϕ3, causing it to output a HIGH (a phase 3 clock enable). The second CP1 pulse also toggles the even phase enable flip-flop to its set state.

(4) One-half period later, the second CP0 pulse appears and, in conjunction with the high from the set output of the even-phase enable flip-flop, will enable AND gate ϕ4, causing it to output a HIGH (a phase 4 enable). The second CP0 pulse also toggles the odd-phase enable flip-flop to its clear state.



CI 85-3432
17e

Figure 5–17. Four-phase master clock.

Assuming there are no malfunctions and the circuits remain energized, this process will be continuously repeated. Figure 5-17,B, is a timing diagram illustrating these relationships.

Exercises (269):

1. What is the purpose of the free-running multivibrator in a single system?

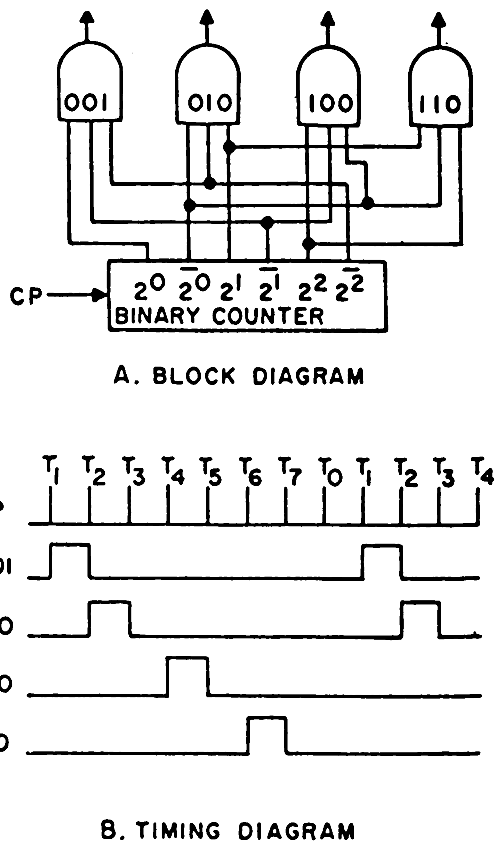
2. What is the output of the pulse shaper in a single-phase system used for?

3. What is the biggest disadvantage of a single-phase system?

4. Refer to text figure 5-16. What is the relationship of $\phi 1$ and $\phi 2$?

5. What is the purpose of using a multiphase system?

6. Refer to text figure 5-17. When will the enable flip-flop change states?


001
010
100
110
CP
2^0 $\overline{2^0}$ 2^1 $\overline{2^1}$ 2^2 $\overline{2^2}$
BINARY COUNTER
A. BLOCK DIAGRAM
T_1 T_2 T_3 T_4 T_5 T_6 T_7 T_0 T_1 T_2 T_3 T_4
CP
001
010
100
110
B. TIMING DIAGRAM

CI 85-3433
17e

Figure 5–18. Binary counter as timing pulse generator.

5-5. Timing Distribution and Control Logic

Timing distribution is normally accomplished through the use of binary counters and ring counters. We will discuss both in this section.

270. Specify the use of binary counters for timing distribution and their operation.

Binary Counters. Binary counters may be used to produce a controlled sequence of pulses. These pulses may, in turn, be used to enable circuits which will cause the operation necessary for program execution to occur in logical sequence.

This concept may be illustrated by the circuit in figure 5-18,A. As indicated by the timing diagram, figure 5-18,B, this circuit will produce a sequence of four timed pulses per clock cycle. More specifically, circuit arrangement is such that each output will be set true by a specific clock pulse during each clock cycle and will be set false by the next subsequent clock pulse. For example, consider the output of AND gate 001. It goes true at time T1 of the first clock cycle and false at time T2. It goes true again at time T1 of the second clock cycle and goes false at time T2. This action continues for as many clock cycles as necessary to complete program execution. In like manner the output of AND gate 010 will go true at time T2 of each clock cycle and false at time T3; the output of AND gate 100 goes true at time T4 and false at time T5; and the output of AND gate 110 goes true at time T6 and false at time T7. These outputs may then be used to enable and disable circuits as necessary for proper program execution.

Exercises (270):

1. Why would binary counters be used for timing distribution?

2. What may the output of a binary counter be used for?

3. Refer to text figure 5-18. At what time interval will AND gate 010 go true?

4. Refer to text figure 5-18. How many timed pulses will be generated by each clock pulse?

271. State the use of ring counters and timing chains for timing distribution and their operation.

Ring Counters and Timing Chains. In computer parlance, a "sequence" is a means of executing an instruction as a series of logical commands separated in time. The master clock outputs initiate and control the operation of a form of ring counter called a "timing chain." Although the specific function of a given sequence may vary from one computer to another, a computer will determine the types of operations to be performed and then select the appropriate sequence or sequences. A typical computer sequence might be:

(1) The A sequence reads the next instruction from memory; determines whether jump, stop, or short conditions have been met; and selects the next sequence.

(2) The B sequence acquires the contents of the index register to allow for operand modification if specified by the instruction.

(3) The C sequence performs arithmetic and logical operation.

(4) The D sequence acquires or disposes of the operand according to the requirements of the current instruction.

Assume that the timing chain presented in block diagram form in figure 5-19,A is one of the timing chains in the typical computer mentioned above. Assume, further, that during the execution of the A sequence it becomes necessary to clear a particular register and transfer data to that register from another register. As a particular timing chain flip-flop (in this case, assume T03) is set, a command enable signal will be sent to clear and load the receiving register. On phase 1 (at T11), the clear command will be fully enabled and the register cleared. On the next phase (phase 3 at time T13), the loading gates will be fully enabled and the register loaded. Figure 5-19,B illustrates the timing relationships.

Consider now the representative timing chain in figure 5-20. This circuit will execute on sequence of operations if function #1 gates are enabled and another sequence of operations if function #2 gates are enabled.

Initially, the timing chain flip-flops must be set to their states by applying a high on the master clear (M.C.) line. Then, if function #1 enable is applied:

(1) The LOW function #1 enable is inverted to a HIGH by the inverters and will partially enable AND gates 3N11, 3N21, and 3N23, and 3N31.

(2) The HIGH at the clear output of the run flip-flop is passed through the inverters (that is, inverted to a LOW and then back to a HIGH by the inverters) to partially enable AND gate 2T02. When the LOW timing chain enable is applied, it is inverted to a HIGH by the inverter and will fully enable AND gate 2TO2, causing it to output a HIGH.

(3) The HIGH from AND gate 2T02 will fully enable OR gate 2T01, causing it to output a HIGH. This HIGH is passed through the inverters to precondition flip-flop T03 so that when the first clock phase 3(ϕ3) appears, T03 will be toggled to its set state. (NOTE: When T03 is set, the timing chain enable is no longer needed and may be disabled or turned off.)

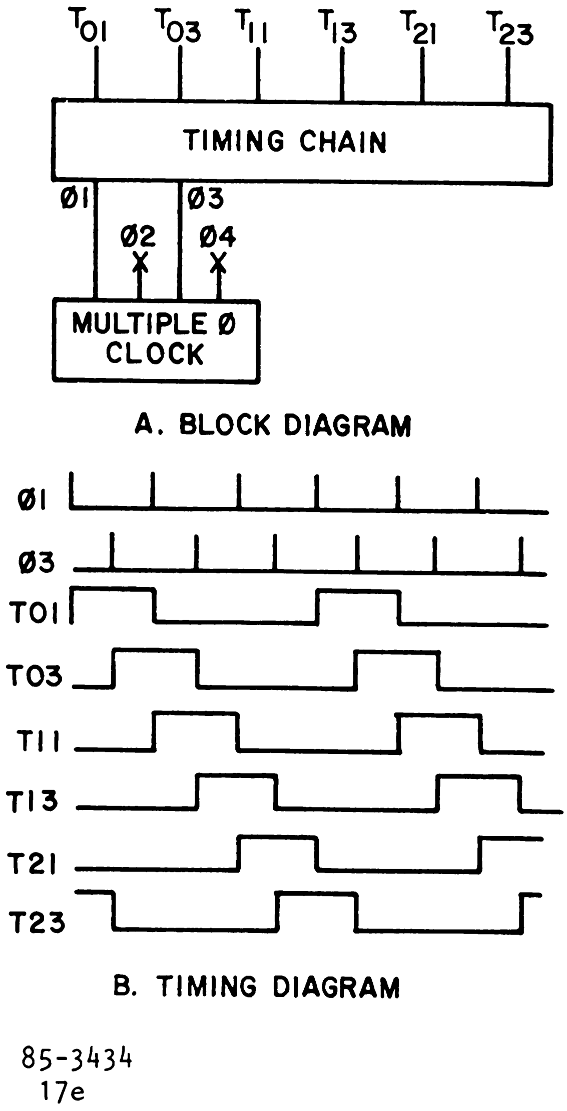
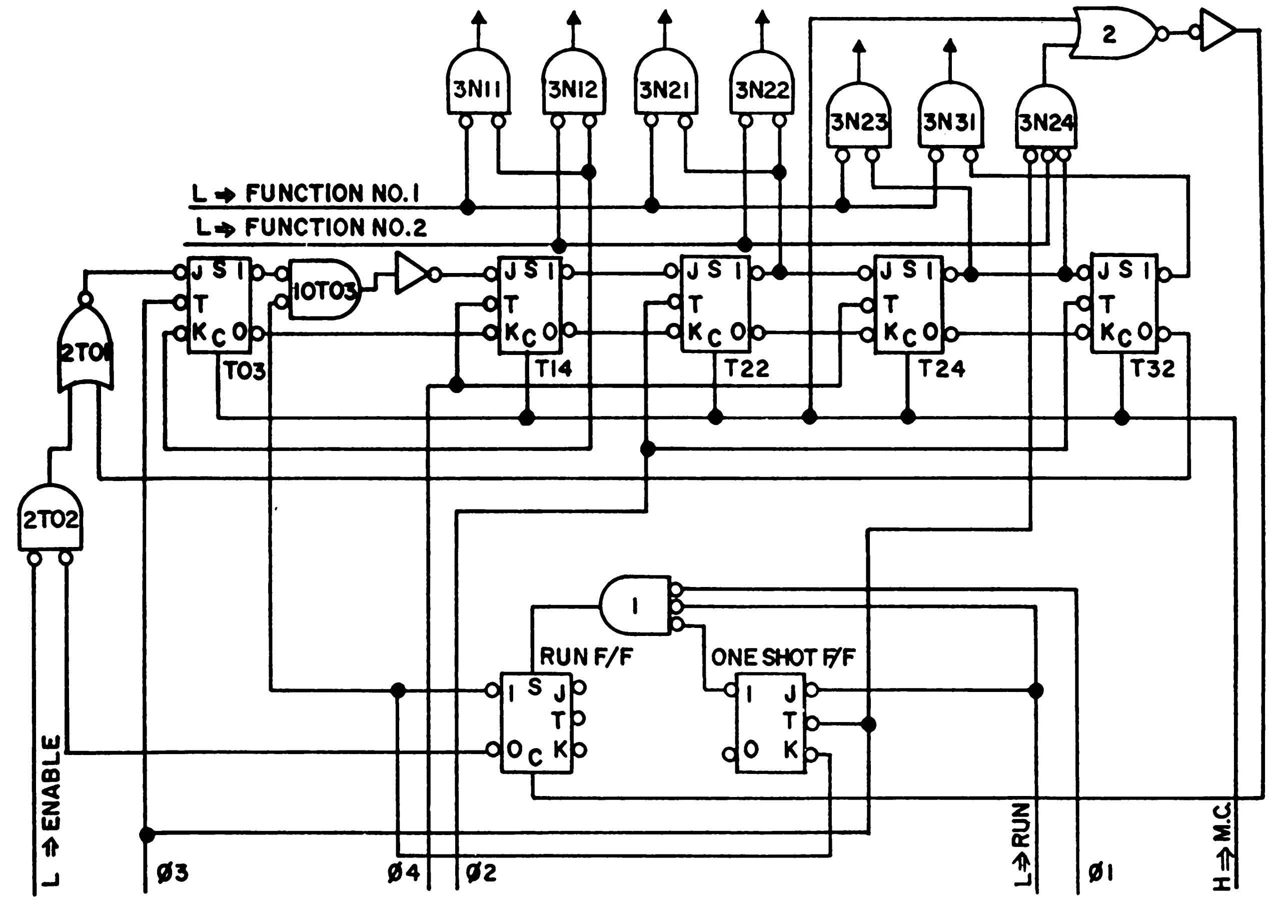


Figure 5–19. Ring counter (timing chain).

(4) The HIGH now appearing at the set output of T03 is passed through the inverters to partially enable AND gate 10T03. (NOTE: The count cannot be advanced beyond this point unless the run flip-flop is set, thereby applying the other enable for AND gate 10T03).

(5) When the LOW run enable is applied, it will be inverted to a HIGH by the inverter to partially enable AND gate 1. It is also inverted to a HIGH to precondition the one-shot so that when the next clock phase 3 appears, the one-shot will be toggled to its set state.

(6) The HIGH from the set output of the one-shot is passed through the inverters and will partially enable AND gate 1. When the next clock phase 1 (ϕ1) appears, AND gate 1 is fully enabled and will output a HIGH which will set the run flip-flop. (NOTE: When the run flip-flop is set,



CI 85-3435
17e

Figure 5–20. Representative timing chain.

the run enable is no longer needed and may be disabled or turned off.)

(7) The HIGH from the set output of the run flip-flop is passed through the inverters to fully enable AND gate 10T03, causing it to output a HIGH. The HIGH from the set output of the run flip-flop is also passed through the inverters to precondition the one-shot so that when the clock phase 3 appears, the one-shot will be toggled to its clear state. The low from the clear output of the run flip-flop is passed through the inverters to disable AND gate 2T02.

(8) The HIGH from AND gate 10T03 is inverted to a LOW by the inverter/amplifier and back to a HIGH by the inverter to precondition flip-flop T14 so that when the next clock phase 4 ($\phi 4$) appears, T14 will be toggled to its set state.

(9) With T14 set, several things will happen. The HIGH from the set output of T14 will be passed through the inverters to act as partial enables for AND gates 3N11 and 3N12 and to precondition T03 so that when the next clock phase 3 appears, T03 will be toggled to its clear state. When T14 is set, its outputs also precondition T22 so that when the next clock phase 2($\phi 2$) appears, T22 will be toggled to its set state.

(10) With function #1 enabled, the HIGH from the set output of T14 will fully enable AND gate 3N11, causing it to output a HIGH. The HIGH from 3N11 will, in turn, enable certain gates and logic circuits within the computer, thus setting in motion such operations as are necessary for proper program execution. (NOTE: Since function #2 is not enabled, AND gate 3N12 will at this time remain disabled.)

(11) When T22 is set, the HIGH at its set output will be passed through the inverters to partially enable AND gates 3N21 and 3N22. The outputs of T22 will also precondition T24 so that when the next clock phase 4 appears, T24 will be toggled to its set state.

(12) When function #1 is enabled, the HIGH from the set output of T22 will fully enable AND gate 3N21, causing it to output a HIGH. This HIGH is utilized within the computer to enable and disable gating and logic circuits, thus permitting certain operations previously begun to be terminated and others to be started. (NOTE: Since function #2 is not enabled, AND gate 3N22 will not at this time be enabled.)

(13) When T03 is toggled back to its clear state, AND gate 10T03 will be disabled, causing it to output a LOW.

This LOW and the HIGH from the clear output of T03 will, when passed through the inverters, precondition T14 so that when the next clock phase 4 (the same one that sets T24) appears, T14 will be toggled back to its clear state.

(14) When T24 is set, the HIGH from its set output will, when passed through the inverters, act as partial enables to AND gates 3N23 and 3N24. When T24 is set, its outputs will also precondition T32 so that when the next clock phase 2 appears, T32 will be toggled to its set state.

(15) When T14 is cleared, its outputs will disable AND gate 3N11 and 3N12 and will precondition T22 so that the next clock phase 2 (the same one that sets T32) will toggle T22 to its clear state. The LOW at the set output of T14 is passed through the inverters to partially prepare T03 for reinitiation of the timing chain.

(16) When function #1 is enabled, the HIGH from the set output of T24 will fully enable AND gate 3N23, causing it to output a HIGH. This HIGH is utilized within the computer to enable and disable gating and logic circuits, thus permitting certain operations previously begun to be terminated and others to be started.

(17) When T32 is set, the HIGH from its set output is passed through the inverters to act as a partial enable to AND gate 3N31. Since function #1 is enabled, AND gate 3N31 is fully enabled at this time, causing it to output a HIGH. This is utilized within the computer to enable and disable gating and logic circuits, thus permitting certain operations previously begun to be terminated and others to be started. The LOW from the clear output is inverted to a HIGH, thus enabling OR gate 2T01, causing it to output a HIGH. This HIGH is passed through the inverters to precondition T03 so that when the next clock phase 3 appears, T03 will be toggled to its set state, thus reinitiating the timing chain.

(18) When T22 is cleared, its outputs will disable AND gates 3N21 and 3N22 and will precondition T24 so that when the next clock phase 4 appears, T24 will be toggled to its clear state.

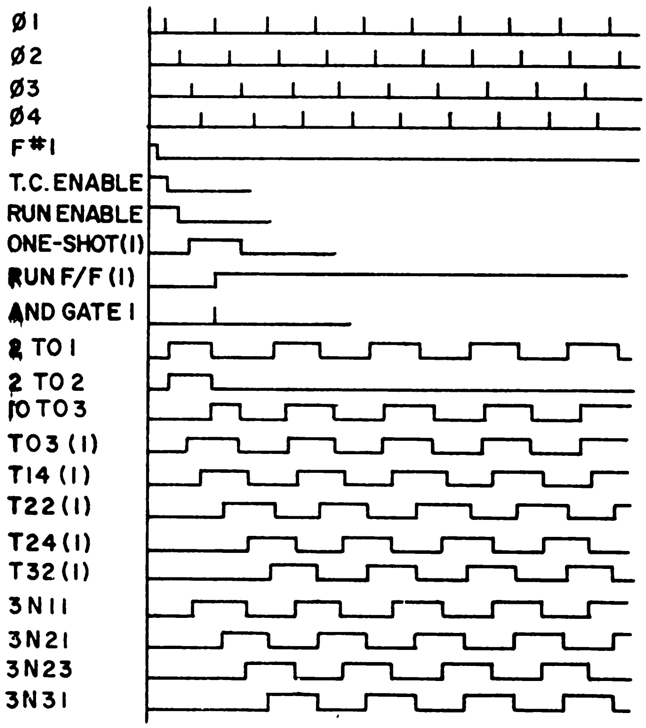
(19) When T24 is cleared, its outputs will disable AND gates 3N23 and 3N24 and will precondition T32 so that when the next clock phase 2 appears, T32 will be toggled to its clear state.

(20) When T32 is cleared, its outputs will disable AND gate 3N31 and OR gate 2T01.

With function #1 enabled the timing chain will continue to recycle until the master clear is again enabled. Figure 5-21 illustrates the timing relationships existing with function #1 enabled.

With the exception that different gates are enabled, operation will be the same with function #2 enabled up to T24 time. At T24 time AND gate 3N24 is partially enabled, and on the next clock phase 3 it will be fully enabled, causing it to output a HIGH. This HIGH fully enables OR gate 2, causing it to output a HIGH. The HIGH from OR gate 2 is passed through the inverter amplifier combination to clear the run flip-flop.

The same clock phase 3 that enables AND gate 3N24 reinitiates the timing chain, which then continues to run until it completes the second cycle. A third cycle, however, will not be initiated since the LOW now present at the set output of the run flip-flop, when passed through the inverters, will disable AND gate 10T03.



CI 85-3436
17e

Figure 5–21. Timing relationships with function No. 1 enabled.

Exercises (271):

1. What is a sequence?

2. How is timing achieved?

Refer to text figure 5-20 to answer the following questions.

3. What initially must be done in order for the circuit to function in proper sequence?

4. What signal fully enables AND gate 2T02?

5. When is the timing chain enable signal no longer needed?

6. When will the one-shot flip-flop be toggled to the set state?

7. What signal will fully enable AND gate 3N11 causing its output to be a HIGH?

8. What will happen when T14 is cleared?

9. What signal reinitiates the timing chain to run until it completes the second cycle?

10. Can a third cycle be initiated?

Answers for Exercises

CHAPTER 1

Reference:

See Module 10005, Unit 1, for answers to objectives 200–219.

CHAPTER 2

See Module 10005, Unit 2, for answers to objectives 220–243.

CHAPTER 3

See Module 10005, Unit 4, for answers to objectives 244–259.

CHAPTER 4

260 – 1. 8_{10}.
260 – 2. 2_4.
260 – 3. 9750_{10}.

261 – 1. 8.
261 – 2. a. 014_8.
b. 023_8.

262 – 1. $\begin{array}{r} 103_8 \\ -12_8 \\ \hline \end{array}$
262 – 2. 000 000 000 111 001.
262 – 3. 71.
262 – 4. It is necessary only to reverse the state of all flip-flops.
262 – 5. There is not a smooth transition through zero.
262 – 6. The false carry is generated when a negative zero is sensed; when it is added to the sum, a positive is produced.

263 – 1. It takes the contents of the two storage registers and combines them in a third register.
263 – 2. The accumulator, or A register.
263 – 3. 37777.
263 – 4. Complemented.
263 – 5. Radix.

263 – 6. $\begin{array}{lr} \boxed{1} & 00373_8 \\ & 00562_8 \\ \hline & 77611_8 \\ \rightarrow & -1 \\ \hline & 77610_8 \end{array}$

263 – 7. Borrow. (This false borrow adds 100000_8 to the minuend.)
263 – 8. Subtractive.
263 – 9. As a positive octal number.

264 – 1. The Q-register holds the multiplier and acts as a down counter. It will count backwards one digit at a time.
264 – 2. The multiplier is right- (or left-) shifted.
264 – 3. To delay the addition for one word length to compensate for the difference in word length between the accumulator and the X-register inputs.
264 – 4. To place the multiplier in the Q-register.
264 – 5. The A- and Q-register.
264 – 6. In the Q-register.
264 – 7. I; M.
264 – 8. Third.

265 – 1. Division by repetitive subtraction. It is time-consuming.
265 – 2. The inspection method (similar to the manual longhand method).
265 – 3. After the Q-register has been increased by 1, the AQ Register is left-shifted one position.
265 – 4. A; Q.
265 – 5. Bit; 1 bit.

CHAPTER 5

266 – 1. The correct choice is A.
266 – 2. To produce a sharp spike.
266 – 3. Used as an input to a Schmitt trigger.
266 – 4. An integrator circuit.

267 – 1. CR1 is shorted, allowing the positive portion of the input signal to be developed across R1.
267 – 2. Choice B is the correct output.
267 – 3. The diode's anode is connected to the capacitor.
267 – 4. A DC restorer.

268 – 1. Squaring off a rounded wave and voltage-level sensing.
268 – 2. Q1 will conduct.
268 – 3. As long as the input pulse exceeds the threshold voltage.

269 – 1. It generates the necessary pulses.
269 – 2. To enable and disable circuits in proper sequence to execute the computer program.
269 – 3. Limited use.
269 – 4. They are generated on alternate half cycles of the astable multivibrator signal.
269 – 5. Permits functions involving more than one operation to be completed during a given clock cycle, or a given operation to be extended over more than one clock cycle.
269 – 6. On the trailing edge of the toggle pulse.

270 – 1. To produce a controlled sequence of pulses.
270 – 2. To enable and disable circuits as necessary for proper program execution.
270 – 3. At time T2.
270 – 4. Four.

271 – 1. A means of executing an instruction as a series of logical commands separated in time.
271 – 2. By utilizing the master clock outputs to initiate and control the operation of a form of ring counter called the timing chain.
271 – 3. The timing chain flip-flops must be set to their clear states.
271 – 4. LOW timing chain enable signal.
271 – 5. When T03 is set.
271 – 6. When there is a LOW run enable signal, phase 3 signal, and toggle signal.
271 – 7. The HIGH from the set output of T22.
271 – 8. AND gates 3N22 and 3N12 will disable, and T22 will be preconditioned so that the next clock phase 2 will toggle T22 to its clear state.
271 – 9. The same clock phase 3 that enables AND gate 3N24.
271 – 10. No, because AND gate 1 10T03 will be disabled.

AUGAFS, AL (860157) 400

☆ U.S. GOVERNMENT PRINTING OFFICE: 1985--631-028/20600 Region #4